ENABLING ONE-HANDED INPUT FOR WEARABLE COMPUTING

A Dissertation
Presented to
The Academic Faculty

By

Gabriel Reyes

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Computer Science
School of Interactive Computing
College of Computing

Georgia Institute of Technology

August 2017

ENABLING ONE-HANDED INPUT FOR WEARABLE COMPUTING

Approved by:

Dr. W. Keith Edwards, Co-Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Gregory D. Abowd, Co-Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Thad E. Starner
School of Interactive Computing
Georgia Institute of Technology

Dr. John T. Stasko
School of Interactive Computing
Georgia Institute of Technology

Dr. Omer T. Inan
School of Electrical Engineering
Georgia Institute of Technology

Dr. Kent Lyons
Research & Innovation Group
Technicolor Research

Date Approved: April 24th, 2017

“If I have seen further, it is by standing on the shoulders of giants.”

Isaac Newton, 1676

To Rylan and the next generation of inventors and explorers

out in the world pursuing and building your dreams... never give up.

ACKNOWLEDGEMENTS

As I reached the end of my Master's degrees, I decided to pursue the Ph.D. as a personal challenge and an opportunity to deepen my technical knowledge in computer science, electrical engineering, and human–computer interaction. I knew it would be a long and exciting journey. I also knew I wouldn't be able to make the trek alone.

I am confident I made the right decision to attend Georgia Tech. I am also extremely fortunate for the opportunity to have been surrounded by an amazing group of people over these past few years.

First of all, I would like to thank my advisors, W. Keith Edwards and Gregory D. Abowd, for welcoming me into their labs, guiding me, working with me to tackle interesting problems, and providing their unwavering support. Keith and Gregory shared their time, ideas, financial support, and connected me with opportunities to excel. Most importantly, they offered friendship and always believed in me. I am forever grateful to both of you.

In addition to my advisors, committee members and mentors inspired me, encouraged me, and willingly expressed their ideas to strengthen my research. Thad Starner shared his wisdom and challenged me to think creatively and critically about the user experience and the exciting field of wearables. I am fortunate to have you as a mentor and teacher. John Stasko stuck with me from the qualifying exam to the defense, and was always a voice of support and common sense. Omer Inan grounded my work and always pushed to uncover the underlying phenomena in the signal. Rosa Arriaga, Agata Rozga, Thomaz Ploetz, and Kaya De Barbaro advocated for me and provided new perspectives to think critically about my work and its applications.

To everyone in the Pixi, Ubicomp, and CCG labs who contributed to my research and life in Atlanta. Special thanks to my fellow senior PhD friends and mentors James Clawson, Edison Thomaz, Craig Tashman, Jennifer Stoll, Yi Han, Lana Yarosh, TJ Yun, Nazneen,

Hwajung Hong, Fatima Boujarwah, Caleb Southern, Marshini Chetty, and Mario Romero. You have paved the way, and it was inspiring to see your energy and work with you. I will also cherish the time spent and memories I have with my other labmates and GT friends, including Abdelkareem Bedri, Himanshu Sahni, Jung Wook Park, Ravi Karkar, Ceara Byrne, Nivedita Arora, Ivan Riobo, Briana Morrison, Cristina Masden, Catherine Grevet, Hayley Evans, Malcolm Haynes, Caitlyn Seim, Joelle Alcaidinho, Sonal Sukheeja, Giancarlo Valentin, Weiren Wang, Cheng Zhang, Hyeokhyen Kwon, and many others. To Aman Parnami for being the brother I never had, along with Ramik Sadana my wonderful friend and travel buddy, Dingtian Zhang my Whoosh co-conspirator and fellow intern explorer, and Chad Stolper my intramural buddy and always wonderful friend to me and my family.

To all the students I have had the opportunity to mentor, learn from, and collaborate with. You are all a major part of this dissertation. I am grateful for your friendship and contributions to this work. Special thanks to Jason Wu who is the only student who actively worked with me on all three projects. Big thanks to: Tara Ramanan, Christopher Chow, and Rushil Khurana on Thumby; Bailey Bercik, Richard Li, Sarthak Ghosh, and Pratik Shah on Whoosh; and Nikita Juneja and Maxim Goldshtein on SynchroWatch. Finally, Victor Chen, Kshitish Deo, and Aaron Quek on other projects.

I must remember the support many staff members provided me behind the scenes. In particular, I would like to thank Scott Gilliland, Sean Brennan, Stephanie Tofighi, Jessica Celestine, Cynthia Bryant, Vivian Chandler, Renee Jamieson, Chanel Bridges, and most of all Monica Ross for all your assistance.

Outside of Georgia Tech, Shwetak Patel was part of my proposal committee and provided guidance and support throughout my time in graduate school. I had the opportunity to intern with Kent Lyons in summer 2016. I will remember our conversations about life, the technology industry, and wearables. I thank you for all your help making this work better. At Microsoft Research, Paul Dietz is one of the most creative and kind people I

have ever met and worked with. You have always been a wonderful friend and mentor, and I consider myself lucky to have met you on this journey. Finally, thanks to the anonymous reviewers and user study participants for their time and contributions.

I was given the opportunity to pursue research early as an undergraduate, and this is one of the many reasons why I decided to pursue graduate research. I would like to acknowledge Jenshan Lin, Wenhsing Wu, Changzhi Li, friends at RFCSR, Sumi Helal, Andy Li, Eric Schwartz, and Antonio Arroyo at the University of Florida. Also, thank you to my summer Research Experience for Undergraduate (REU) mentors Ruyan Guo, Hongbo Liu, Hung Tao Shen, Hayley Shen, and the friends I made along the way.

I appreciate the many sources of funding that made this work possible, from the Georgia Tech Presidential Fellowship, Intel GEM Fellowship, Georgia Tech Goizueta Fellowship, to the GVU Center, and others. All three projects in this dissertation and my tenure as a graduate student were supported by a Google Ph.D. Fellowship. My advisor W. Keith Edwards also supported me with a Google Faculty Research Award. Thanks to Samsung Research, Fitbit Research, Microsoft Research, Intel Corporation, Texas Instruments, Philips Invivo, Technicolor Research, and all my internship mentors for the experience.

I want to thank those of you who have known me for years. My closest friends, you know who you are. Heartfelt thanks to my parents, my sisters, the Jackson family, and the rest of my relatives for always being there for me. My mom, dad, and sisters have shared their love and unrelenting support. They also stopped asking when I was going to finish the PhD a few years ago. I am very glad to share the news, I did it.

Finally, biggest thanks to my wife Kate, who has been my partner and love through it all. I could not have done this without you every step of the way. Your resilience, dedication to your work, and care for me and our son is inspiring. Your love has lifted me up and helped me succeed. My hope is that I have been able to do the same for you. I couldn't imagine life without you and Rylan. Thanks to the gift of being a parent with you, I get to see the world again every day through a new pair of eyes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

A new evolution of computing is emerging around wearable technologies. Wearable computing has been a topic of research for years. However, we are beginning to see adoption by consumers and non-researchers due to advances in embedded and mobile software systems, low-power microprocessor design, wireless technologies, and low-cost sensors. There are still a number of open research challenges in wearable computing, from providing continuous battery power, simplifying on-body networking, addressing privacy and social issues, to designing the interaction experience. Traditional desktop and mobile input technologies, such as mice, keyboards, and in some cases touchscreens are no longer suitable for wearable computing scenarios. In its most common embodiment, wearable computing today relies on a very restricted set of input and output modalities, making this an exciting research area with opportunities for innovation.

The goal of my work is to envision new user experiences and enhance the richness and quality of input modalities available to mobile and wearable computer systems. In this dissertation, I articulate an alternative approach to interaction with computing systems that is specifically focused on wearable, one-handed input techniques. I utilize the smartwatch as a platform of choice for sensing and computation. However, these techniques may also be embedded into other types of wrist-worn devices such as bracelets or fitness bands. The interaction techniques I describe in this dissertation are designed purposefully to eliminate the need to directly interact with the smartwatch touchscreen. I take advantage of the dexterity of the arm, hand and fingers around the smartwatch for gestural interactions. I also leverage the malleability of humans' vocal resonant system to produce non-voice acoustic sounds as input when bringing the smartwatch close to the mouth.

In summary, I present research work in the design, implementation, and evaluation of three systems: (1) Thumby — a gloveless, inertial-based technique that combines the smartwatch with sensors mounted on the thumb to sense wrist and thumb movements and

enable a broad set of finger-level gestures; (2) SynchroWatch — a one-handed interaction technique that tracks the synchronous and rhythmic extension and reposition of the user's thumb (augmented with a passive magnetic ring) through correlation with on-screen blinking controls and without requiring calibration; and (3) Whoosh — an interaction technique that allows a person to control their smartwatch through non-voice acoustics (blowing, exhaling, shushing, sipping, etc.) detected using the device microphone and machine learning.

My hope is that the one-handed interfaces detailed in this dissertation will contribute insights on sensing techniques and interactions to the mobile input and wearable computing research communities, and ultimately reach wearable computing consumers. Through a process of sensor explorations, rapid prototyping, system evaluation and empirical studies, I demonstrate it is feasible to build a set of wearable systems that take advantage of novel on-body sensing and the human anatomy to enable robust and reliable one-handed input. I demonstrate that the set of wearable devices I have built are quickly accessible, may operate in near real-time, and are available when needed. More importantly, these wearable devices enable users to provide useful input, in the form of various discrete and continuous gestures, to a number of applications, including system controls, responding to notifications, and playing games.

CHAPTER 1
INTRODUCTION

1.1 Preface

Over the past 50 years, the keyboard and mouse have remained the primary input devices for desktop-based computing. However, in his essay on manual input, Buxton articulates how today's computer systems make extremely poor use of the human anatomy and our sensory and motor system [13].

"Imagine a time far into the future, when all knowledge about our civilization has been lost. Imagine further, that in the course of planting a garden, a fully stocked computer store from the 1980's was unearthed, and that all of the equipment and software was in working order. Now, based on this find, consider what a physical anthropologist might conclude about the physiology of the humans of our era? My best guess is that we would be pictured as having a well-developed eye, a long right arm, uniform-length fingers and a "low-fi" ear. But the dominating characteristic would be the prevalence of our visual system over our poorly developed manual dexterity."

One of the challenges that Buxton highlighted at the time of his essay is that interactive devices did not take advantage of our manual dexterity. Over the past 10-15 years, a new form of human–computer interaction has emerged in a touch-based paradigm, spurred by the development and widespread availability of mobile phones. Modern smartphones today are sophisticated computing and sensing platforms, enabling people to communicate with others, access information on the web, listen to music, take pictures, play games, and much more.

Most interactions with mobile devices today use touchscreens as input, and still remain predominantly reliant on the visual system. Most users hold the device in one hand and interact with the other, while some with larger hands are able to interact with a single hand using the thumb. Research and commercial input systems have begun exploring a variety of approaches for finger-based input and typing [65, 121], which generally still require the user to interact with a visually-oriented touchscreen and may suffer from occlusion, or “fat finger" issues [8, 101].

Others take advantage of alternative input channels on the mobile phone for scratching, tapping, whacking, and blowing [38, 41, 53, 93], all of which may be paired with auditory or visual output for feedback. These interactions hint at a possible decoupling of the input location and the type and location of the output feedback. For example, a pair of earbuds could provide auditory feedback while the user interacts with a mobile phone in their pocket. Interestingly, that means that systems can extend the input interaction to other devices or body parts, hinting at what comes next.

A new evolution in computing is emerging around wearable technologies. Wearable computing as the next wave of computing has been the topic of research for years. However, the field is only recently beginning to see adoption by consumers and non-researchers due to advances in embedded and mobile software systems, low-power microprocessor design, wireless technologies, and low-cost sensors. There are still a number of open research challenges in wearable computing, from providing continuous battery power, simplifying on-body networking, addressing privacy and social issues, to designing the human–computer interface [104, 105]. In its most common embodiment, wearable computing today relies on a very restricted set of input and output modalities, making this an exciting research area with opportunities for innovation. Traditional desktop and mobile input technologies, such as mice, keyboards, and in some limited cases touchscreens (e.g., wet hands or sweating) are no longer suitable for wearable computing scenarios.

There are two main categories of wearable devices available today.[1] These devices are either worn on (1) the arm, hand, and fingers, or (2) the head. Devices worn on the arm, wrist or fingers include smartwatches (Apple Watch), fitness trackers (Fitbit Charge), smart bracelets (Ringly Rendezvous), sensing armbands (Thalmic Myo), and smart-rings (Oura Ring). Head-mounted devices include a range of smart glasses that rely on a visual display (Google Glass), are auditory only (Vue Smart Glasses), or are used for audio-video capture only (Snap Spectacles). There are also a broad number of systems for augmented reality (Microsoft HoloLens) and virtual reality (Google Daydream View, HTC Vive, Oculus Rift, etc.) that are outside the scope of this dissertation.

1.1.1 Designing The Interaction Experience

Decoupling the input method from the output feedback is a system design paradigm that is well suited for wearable computing, accommodating the use of disaggregated input and output devices which are especially designed to fit the human body (e.g., head-mounted glasses, gesture input armbands, wireless headphones, and others).

Both the hand- and head-mounted categories of wearable devices share many properties in terms of their input and output capabilities. For example, the most common approach today as output from the system to the human user is a graphical display, either on the wrist or head-mounted worn just in front of the eye. Graphical displays are ideal for presenting text, icons, and other high-resolution information. Auditory output is also provided through loudspeakers for smartwatches or headphones. Bone conduction transducers near the ears may also be used [103]. Haptic feedback is another mechanism of gaining the user's attention through various vibration patterns [64].

For input from the user to the system, the most common modalities are speech and touch. Speech provides a rich, high-bandwidth channel between the user and the system. For touch input, a capacitive touchscreen and hardware buttons are the common input

[1] The devices listed here are not meant to represent an exhaustive coverage of each category, but instead serve as individual examples in each category to inform the reader

modalities. A restricted set of touch inputs, such as taps and swipes, are available on both wrist-worn and head-mounted devices and can be used to navigate through icons and other information presented on a graphical display.

While speech and touchscreens are the primary input modalities to date, there are a number of important downsides inherent in their use. For example, long-form speech may be socially inappropriate in many public places, is not ideal for repetitive microinteractions, and raises privacy concerns since others may overhear your messages and interactions. Spoken commands are sensed by the system using on-device microphones but sometimes may present challenges when used in noisy environments. Occlusion or “fat finger" issues from the mobile phone are exacerbated on wrist-worn devices with small screen sizes. In general, modalities other than speech and touch may be more appropriate in a variety of use cases, and yet have been drastically underexplored by the research community. One such use case is when the person’s second hand is busy or unavailable for interaction. Motion gestures, one example modality that does not require directly interacting with the device, are beginning to gain popularity for simple purposes, such as checking the time or turning the screen on in response to a notification.

The goal of my work is to envision new user experiences and enhance the richness and quality of input modalities available to mobile and wearable computer systems. In this dissertation, I articulate an alternative approach to interaction with computing systems that is specifically focused on one-handed input techniques using wrist- and finger-worn devices using inertial and acoustic sensing. I utilize the smartwatch as the platform of choice for sensing and computation. However, these techniques may also be embedded into other types of wrist-worn devices, such as bracelets or fitness bands.

The wrist is an interesting target location for input for a number of reasons. The physiology and dexterity of the hand and fingers nearby enable a wide variety of gestures. The smartwatch at the wrist can also be moved in 3D space due to the arm’s degrees of freedom; for example, to bring it close to the mouth. The interaction techniques I describe in this

dissertation are designed purposefully to eliminate the need to directly interact with the smartwatch touchscreen using the second hand and support single watch-handed scenarios. I take advantage of the dexterity of the arm, hand and fingers around the smartwatch for gestural interactions. I also leverage the malleability of humans' vocal resonant system to produce non-voice acoustic sounds as input when bringing the smartwatch close to the mouth.

1.2 Thesis Statement

In this dissertation, I develop multiple interaction techniques and algorithms using wrist- and finger-worn sensors to enable one-handed input, and I demonstrate through experimental results the feasibility and robustness of such systems. Specifically, the systems and experiments presented in the following chapters address my thesis statement, which reads as follows:

One-handed input recognized using on-body inertial and acoustic sensing can enable an *expressive*, *fast*, *practical*, and *robust* modality for wearable computing.

The one-handed interfaces detailed in this dissertation seek to contribute novel insights on sensing techniques and interactions to the mobile input and wearable computing research communities. Leveraging a process of sensor explorations, rapid prototyping, and system evaluation through empirical studies, I demonstrate that it is feasible to build a set of practical wearable systems that take advantage of novel on-body sensing and the human anatomy to enable robust and reliable one-handed input. These wearable techniques enable users to provide useful input to a number of applications, including system controls, responding to notifications, and playing games.

1.3 Research Goals

In this dissertation, I demonstrate my thesis empirically with three hypotheses:

- Finger-level gestures can be detected accurately and robustly using a combination of inertial sensing at the wrist and the thumb, rotation-invariant features, and machine learning. (Thumby)

- Rhythmic movement of the thumb instrumented with a magnetic ring can be quickly, accurately, and robustly correlated to on-screen blinking controls, requiring no system calibration or prior knowledge of sensor placement. (SynchroWatch)

- Non-voice acoustic events for use in interactive applications can be captured by the smartwatch microphone, detected accurately and robustly using machine learning, and extended using a passive 3D-printed watch case. (Whoosh)

The key entities in each of the systems I describe in this dissertation are: the user at the center of all interactions, the interaction technique that provides the user with useful input capabilities, the hardware that enables said interactions, and the software algorithms and analysis that turn input from the user and sensor data into relevant actions in the interface.

My dissertation focuses on addressing the hypotheses above via four research goals. The four descriptors — expressiveness, speed, practicality, and robustness — are framed around relevant properties or metrics of the interaction technique, hardware and form factors, and software algorithms. I approach these research goals using three different interaction techniques as case studies. I define and explain the metrics I use to characterize each of the goals below. The research goals of this dissertation are:

- Goal 1: Achieving *expressiveness* through a spectrum of interaction events and wearable interfaces

- Goal 2: Enabling rapid *speed* of interaction and reducing the time between intention and action

- Goal 3: Expanding the *practicality* of one-handed input techniques and form factors
- Goal 4: Building *robust* detection approaches for multiple users and everyday activities

Card et al. [16] describe *expressiveness* in terms of the number of elements in the input and output set for a device's possible actions. For this work, I will consider the expressiveness of the interaction techniques based on: 1) the number of unique input events supported; 2) information transfer rate [80] calculated in terms of number of events, accuracy, and interactions per second; 3) the purpose of the technique or gesture within the interface (i.e., activation, command, or notification-response); and 4) the type of interface and applications it enables.

Starner discusses the challenges of wearable computing in a two-part article series [104, 105]. One of the challenges he discusses is the design of the human–computer interface. One of the driving design goals of the human–computer interface for wearable computing is the *speed* of interaction and the desire to reduce the time between intention and action [103]. For this work, I will consider the speed of the interaction techniques based on: 1) the time between intention and action, or the time it takes for the user to be ready to provide useful input; and 2) the speed of interaction, or the amount of time it takes to complete the input gesture itself.

Recent work in wearable and ubiquitous computing [1, 85, 104, 105, 108] has focused on building *practical* systems that can be deployed for use in the real world. I will consider the practicality of the interfaces based on metrics that relate to the hardware: 1) the level of instrumentation required to enable the interaction, measured based on the amount and characteristics of the devices and form factors; and 2) the prototype readiness for deployment expressed in terms of an adapted version of the NASA Technology Readiness Levels (TRL) [72].

Finally, deploying an interface across multiple users and everyday activities requires *robust* detection algorithms and analysis approaches. I will consider the robustness based

on: 1) the accuracy of the technique; 2) its usage across multiple users with user-dependent and user-independent considerations; and 3) its support for input during activities with different levels of uncertainty and noise (e.g., global motion of sitting vs. walking, ambient noise for acoustics).

1.4 Research Approach

To improve the quality of wearable human–computer interfaces, I have built and designed several embodiments of systems that enable expressive input for applications on-the-go. The main driving force behind building system explorations is to develop better understanding of how, as developers and designers, we can better tailor input approaches to fit the human experience, in particular for one-handed scenarios. This dissertation explores an end-to-end approach to developing a set of novel techniques for one-handed input.

My research approach follows a user-centered design process that begins with motivating the needs of an everyday user of wearable computing and the various scenarios in which they might require input. To begin understanding the space, I conducted an extensive review of related work in wearable computing and novel on-body input devices (see Chapter 2 and Chapter 3) that hints at some of the possibilities of new forms of techniques and applications, shifting away from a phone-centric interaction paradigm. I then asked myself the question, is it feasible to design and develop wearable interaction techniques that enable robust and reliable one-handed mobile input for these scenarios and applications? I conducted an exploration of a variety of sensors (e.g., force sensitive resistors, accelerometers, gyroscopes, electromyography, microphones, and depth cameras) and built a number of prototypes in the laboratory. My exploration of the space through prototyping has informed the choice of sensing technologies and devices described in this dissertation. Based on initial experiences with these prototypes, I narrowed my focus to inertial sensors (accelerometer, gyroscope, and magnetometer) and acoustic sensors (microphone).

The inertial sensors and microphones provide sensor data that enables tracking fine

motions of the fingers, arms, and wrists, as well as subtle sounds produced by the mouth. These devices are readily available in many commodity devices, including smartwatches, and require minimal additional instrumentation. Each of the projects described in this dissertation began with initial data collection to assess the feasibility of the technique. Once initial feasibility was in sight, I developed a gesture or event recognition pipeline to collect, pre-process, and detect the given input. I then rapidly iterated on testing and enhancing the pipeline to optimize the system. The next step was to test and evaluate the technique with human subjects and collect additional sensor data to assess robustness and accuracy across participants. Feedback from participants during the evaluation phase further informed the desired applications that could be enabled. Finally, I developed various demonstration applications to highlight the technique in useful scenarios. A modular architecture allows for easy integration of the classification pipeline with applications, edging closer to the vision of always-available computing with a fully integrated sensing and interaction platform on-the-body.

1.5 Acknowledgement of Collaborators

This dissertation presents multiple projects I have driven and led during my Ph.D. tenure. However, completing this research could not have been possible without numerous collaborators who have significantly contributed to the work. Collaborators include my co-advisors, committee members, mentors, labmates and fellow Ph.D. students, Master's and Undergraduate students I have mentored and learned from, and external collaborators. Their contributions are acknowledged in an earlier section of this dissertation and through authorship in publications resulting from our work. However, for the remainder of this dissertation, I use only the first person singular to describe the projects.

1.6 Dissertation Overview

To fully answer my thesis statement, the remainder of this dissertation describes in detail the one-handed input techniques I have developed. I present how the techniques are built and evaluated, in terms of form factor design, interaction design, software pipeline, hardware prototypes, and exposure to and evaluation with human subjects.

Specifically, I investigate the performance of three systems — Thumby, SynchroWatch, and Whoosh — using a series of empirical user studies and system evaluations to understand the user experience and quantify system performance.

In Chapter 2, I summarize research from a number of domains, including wearable computing, novel input devices, wrist-based devices, finger gestures, and other interfaces. I expand on the related work section and present a taxonomy that spans research in one-handed input techniques using the arm, hands, and fingers in Chapter 3. I identify key opportunities for the space of one-handed input and interaction techniques, such as exploring a broad set of thumb gestures, introducing synchrony between the interface and the user, and using the mouth as an actuator.

In Chapter 4, I explore the space of gestures using wrist- and thumb-worn inertial sensors to enable a broad set of one-handed interactions with a system called "Thumby." I present two iterations of the system using sensors worn on the thumb-only and a combination of wrist and thumb. I perform multiple online and offline evaluations with participants to assess the broad range of gestures the device enables.

In Chapter 5, I present "SynchroWatch" which is a subtle interaction technique for wrist-worn devices that leverages synchrony between thumb movement and on-screen blinking controls. The technique uses magnetic sensing on-device along with a passive magnetic ring on the thumb. This work is the first to introduce synchronous gestures for smartwatches using sensors on the device. I evaluate the technique to determine initial parameters and follow up with a user study comparing the performance of SynchroWatch to touch-based

input, which is commonly used on smartwatches today.

In Chapter 6, I discuss an interactive input system called “Whoosh” for smartwatch interactions using non-voice acoustics. The technique allows a person to control their smartwatch by blowing, exhaling, shushing, sipping or puffing on and around the screen. The watch uses its microphone and machine learning to identify the breath patterns of each acoustic event, then assigns an action to each. I evaluate this system offline with two user studies and a total of 18 participants.

I conclude in Chapter 7 with a summary of key contributions and limitations of the interaction techniques presented in this dissertation. I also present several insights on my experiences building interactive systems and the challenges of developing robust and reliable input techniques for scenarios in-the-laboratory and in-the-wild.

CHAPTER 2
RELATED WORK

2.1 Background

In this chapter, I review related work and discuss how it influences the work presented in this dissertation. I begin with a brief history of wearable computing and provide an overview of novel on-body input devices. More specifically, I emphasize the interest in the wrist, arms, fingers, and mouth as ideal human effectors for input and specific target locations for the placement of on-body devices and interactions.

My work aims to build on this prior research by focusing on developing one-handed wearable techniques that enable new forms of always-available input [76]. Always-available mobile input does not require a user to necessarily pick up, hold, or look at a device leading to a more immediate and natural input experience. For example, a smartwatch is readily available at the wrist and can be manipulated using the fingers, thumb, and mouth.

2.2 Wearable Computing

Wearable computing pursues an interface ideal of "a continuously worn, intelligent assistant that augments memory, intellect, creativity, communication, and physical senses and abilities" [104, 105]. Historically, the field of wearable computing has primarily defined itself by key characteristics, such as portability, enabling hands-free use, opportunistically attracting the user's attention, being continuously available, and being context aware [74, 94]. While many systems challenges remain, such as balancing network resources with power requirements, there are also fundamental human concerns that must be addressed, including understanding social norms around the use of these devices, designing for privacy, and creating innovative human–device interfaces.

One advantage of wearables over more traditional mobile devices is that mobile devices are often stored in pockets, purses, or backpacks, requiring users to search for them and remove them in order to access basic functionality. Physically retrieving the device incurs a non-trivial time cost, and research has shown that it can constitute a significant fraction of a simple operation's total time [7]. Thus, in wearable computing, the key advantage is that the user is bringing the devices much closer to their senses. For example, using a head-up display brings the device near the visual and auditory channels. Similarly, a wrist-worn device may also be quickly summoned to the visual field. However, the act of using a wearable computer still demands a high level of visual attention and typically the use of two hands, in the case of the smartwatch. There are many situations where the user's focus and vision is needed elsewhere (e.g., face-to-face conversation, biking, driving, etc.) or when distractions could be dangerous, as well as when the hands are busy (e.g., carrying groceries, manipulating physical objects in the environment, etc.).

Nevertheless, the primary output channel in most computing tasks has traditionally been visual, due to the high bandwidth of this channel. The wearable computing community has used the head-up display as an effective mobile display built into eyeglasses or as an independent unit. Google Glass is one example of a recently available self-contained wearable computer with a head-mounted display. The device displays information using visual and auditory channels and is designed to be controlled using touch and natural language voice commands. A wrist-worn smartwatch also serves as a small mobile display. There are a variety of smartwatches on the market (e.g., Sony Smartwatch 3, Apple Watch, Samsung Gear, and others). Other visually-based wearable interfaces have been explored as well. For example, Saeedi et al. have created initial prototypes of contact lens displays which still require many more advances before they are ready for commercial applications [96]. The MicroOptical SV-6 was a lightweight head-up display viewer, popular for use in a number of industrial and medical applications.[1]

[1] `http://www.cruxial.com.sg/Microoptical/vga/SV6mobile_MK-0061A.pdf`

On-body projection is another approach to quickly accessing information. Researchers have explored projecting onto a person's forearm, hands, and in the environment around the user [39, 42, 75]. High resolution and high frame rate projection technology is available but it remains too bulky for everyday use. There are also other technical and wearability issues to address, including color distortion from the skin, the presence of hair on a person's arm, and realignment of the projection and interaction surface during daily activities.

Alternate modalities for wearable computing include clothing-based interfaces and e-textiles [11, 83]. One of the main disadvantages of this approach is that all garments we wear every day must be embedded with a variety of sensors and display technology. Additionally, the embedded technology may not be as durable, washable, disposable, flexible, or customizable as a person's clothing typically is. Either a modular solution that can be switched out to be used between garments would be needed or a new material that is washable and durable can be developed [90]. Researchers have also explored alternative audio-only modalities for wearable computing, exploring speech recognition and contextual notifications [100].

2.3 Novel Input Devices

Previous research has explored a variety of novel input techniques to provide new forms of input to wearable and mobile computing systems. For example, the Skinput system leveraged the natural acoustic conduction properties of the arm for input [42]. The system has shown promise in detecting the timing of touch and tap events in which the user targets a location on the arm or fingers for input. The SixthSense project used a camera-worn system around the neck that combines projected information with a color-marker-based vision tracking system [75]; the user interacts with an "invisible" canvas or surface floating in the front of the user's head/chest. This system's major challenge is occlusion of the hands when interacting with objects during real world tasks. The hands-busy nature of this interface impedes the user from interacting with physical objects while the arms are raised

and used for input gestures. Furthermore, there are accuracy limitations using computer vision, as distinguishing finger taps from hovering in the interface is particularly challenging. Other techniques based on computer vision have been explored, which provide wearable gesture recognition in a variety of lighting conditions and mobile settings [106].

Other biologically-mediated signals have been used for input, for example vital signs and galvanic skin response sensing have been used for assessing a user's emotional state [73]. These signals are typically used as implicit input in context-aware applications, as they are not available for direct input and are driven subconsciously. Similarly, sensing technologies including functional near-infrared spectroscopy (fNIR) and electroencephalography (EEG) have been explored to assess cognitive and emotional state [34, 50]. Other brain signals have been used as direct input by users with disabilities [26]. These technologies are generally either extremely low bandwidth, or difficult to control by users. Further advances are needed in these areas.

Other input approaches have taken the form of e-textiles or clothing-embedded computing, in which the input device is worn as a part of one's garments [11]. Some of these have taken the form of wearable gloves [107] which may provide rich input with numerous actions for a user to perform, but these are typically uncomfortable, make it more difficult for the user to interact with objects in the real world, and are disruptive to tactile sensation. As noted before, other "smart fabric" systems embed sensors directly into fabric [89]. On the other hand, smart garments may be ideal for niche applications, such as health and fitness tracking.[2] In contrast, people already wear many different forms of computing including smart watches, cellphones, and health and activity trackers on their wrists, hips, hands, etc., suggesting that such form factors can be instrumented with additional sensing functionalities.

Finally, and most directly related to my work, previous research on wearable input devices has explored how users can interact with a mobile or wearable device without the

[2] `http://www.liveathos.com/apparel/technology`

need of the second hand. GestureWrist is an input device that recognized gestures by capacitively measuring wrist-shape changes and forearm movement, using sensing elements embedded in a normal wristband form factor [92]. Amento et al. placed contact microphones on a user's wrist to assess finger movement and focused on one-handed input [2]. The Hambone system is a bio-acoustic gesture interface and through a Hidden Markov model yielded classification accuracies around 90% for four hand gestures [23]. Starner et al. developed a self-illuminating, wearable, infrared computer vision system to control various home automation tasks [106]. Researchers have also analyzed the electrical signals generated by muscle activity using electromyography (EMG) and this approach is particularly good at sensing muscle exertion for continuous variation of input controls [99]. This approach requires signal amplification and typically the application of conductive gel for effective signal acquisition; however, advances in dry electrode systems and the availability of low-noise, analog front-end analog-to-digital converters and amplifiers have made this approach suitable for commercialization and everyday use.[3] Tongue input has also been explored by a number of researchers and is workable for low-bandwidth, discrete events [54, 98]. Finally, the handheld Twiddler keyboard has been used to achieve text input rates of 60wpm with sufficient training [71].

2.4 On-Body Sensing and Interaction Techniques

In this section, I present work related to on-body sensing and interaction techniques. Primarily, I focus on interactions with sensing and devices placed on the arm, wrist, hand, and fingers.

2.4.1 Arm and Wrist-Based Sensing Techniques

Several custom-hardware solutions sense the surrounding surface area of the wrist and forearm. Prior work includes approaches with bio-acoustics [2, 23, 42] using wrist- and

[3] https://www.thalmic.com/

forearm-worn sensors and enabling various tapping and striking gestures. Forearm electromyography (EMG) [99] has also been explored in research and commercially over the past few years. Rekimoto [92] used capacitive sensing to detect changes in the wrist contour and detect hand poses. WristFlex [22] used an array of force sensitive resistors worn around the wrist to capture its deformation, enabling the interface to distinguish subtle finger pinch gestures. Tomo [123] used electrical impedance sensing with a wrist-worn array of electrodes to capture gross hand gestures and finger pinches. Perhaps most similar to SynchroWatch in its gesture design, ThumbSlide [3] captured the sliding movement of the thumb along the hand by detecting changes in the wrist contour using an array of photoreflectors. Kim et al. [60] used a wrist-worn camera to detect various finger poses and gestures using a kinematic model. These solutions are capable of providing a diverse input vocabulary, but may suffer from limitations such as bulkiness of hardware, occlusions for line-of-sight solutions, cost and complexity.

2.4.2 Finger-Based Sensing

Others have proposed augmenting the fingers or designing custom form factors (such as rings) to capture one-handed finger and thumb movement. DigitSpace used a pair of magnetic-sensing nail-ring chains on the fingers to detect the movement of a magnet-augmented thumb. Rings have been used to detect a variety of input gestures using magnetic sensing [4, 18], infrared reflection [81], optical sensors [77, 118], bend sensors and accelerometers [110], and a combination of inertial and optical sensors [59]. To highlight a few examples, ThumbRing [109] proposed an active ring with an inertial measurement unit worn on the thumb to track motion, enabling users to touch and slide on their fingers. ThumbRing evaluated their technique while the user is stationary and walking but does not presented a generalized solution that is scalable across scenarios and is rotation-invariant. uTrack [18] went beyond discrete input and introduced a technique for 3D tracking of a thumb magnet using a pair of magnetometers on the back of the fingers.

2.4.3 Data Gloves

Data gloves have been in use for a long time and represent one of the most important efforts aimed at acquiring hand movement data, detecting very fine gestures such as finger movement, and modeling of the entire human hand [107]. The Acceleration Sensing Glove (ASG) placed one 2-axis accelerometer on the back of the hand as a tilt motion detector for moving the pointer on the screen and three other 2-axis accelerometers were placed on the thumb, index finger, and middle finger to operate as mouse click buttons [87]. Hrabia used a glove with embedded motion sensor boards over every finger for precise modeling of the whole hand [51]. Glove solutions for detecting thumb-to-phalange events include KeyGlove[4], Figure-Joint Gesture Wearable Keypad [30], and Thumbcode [91]. My work emphasizes a gloveless approach to solving interaction challenges. While glove-based systems can potentially provide access to a rich range of gestures, they are typically bulky, require wiring to a separate processing unit, and—most detrimental to day-to-day use—also affect the haptic sensation during finger gestures.

2.5 Smartwatch-Based Interactions

Others have proposed augmenting the watch or designing custom watch form factors to capture input on and around the device.

The most common way of interacting with a smartwatch device is using the second hand for touchscreen input or button controls. Speech recognition is a secondary modality enabling high bandwidth dictation. While a number of projects have focused on expanding the input capabilities on smartwatches for two-handed input using around-device sensing and custom form factors [40, 79, 116, 117, 122], my work focuses on one-handed (the hand wearing the watch) discrete interactions on a commodity smartwatch.

In particular, one-handed interactions support situations in which the user's other hand may not be available if the person is carrying something or suffers from a disability. A few

[4] `http://www.keyglove.net/`

techniques have motivated the use of commodity sensing on a smartwatch and inspired the design of Thumby and SynchroWatch in this dissertation. Serendipity [114] uses motion sensors in a smartwatch to distinguish fine-motor gestures such as pinching, rubbing and tapping fingers. A tilting metaphor is another approach that has been used for menu navigation, selection, and text entry [33, 35]. The final system in this dissertation —Whoosh— uses the microphone in the smartwatch to detect non-voice acoustics with the mouth as an actuator for no-handed input [93].

Prior work also focuses on developing custom watch devices. Facet provides a multi-display wristband consisting of multiple independent smartwatch screens, enabling a rich set of touch input techniques [69]. Xiao et al. provide a multi-degree-of-freedom mechanical watch face that supports continuous 2D pan and twist, as well as binary tilt and click [117]. Laput et al. use proximity sensing [63] to enable 'skin buttons' that can be activated by touching the skin around the watch. Oakley et al. use proximity sensing around the watch face to capture interaction on the edge of small devices [79]. WatchIt introduces a custom watch band for eyes-free interaction [88]. While modifying the case or band around a smartwatch with electronics may provide additional interaction capabilities, it will also place varying degrees of power constraints on a device with limited battery capacity.

In Chapter 6, I present the use of a passive 3D-printed watch case — dubbed "FluteCase" — to increase the expressivity of the event set, with no additional demands on battery or computation. While the technique does require an active microphone and continuous analysis, the majority of smartwatches today are already "always-on" for hotword detection (e.g., "Ok Google"). It could be possible to modify this device functionality to include recognition of Whoosh events.

2.6 Summary

The interfaces described in this dissertation build on and extend the related work described in this chapter. The goal is to use one-handed interaction techniques to enable always-

available input in everyday computing scenarios. The techniques can serve to replace or supplement current interaction techniques available on commodity wrist-worn devices, and in particular are meant for fast and subtle microinteractions.

CHAPTER 3
ONE-HANDED INPUT TAXONOMY AND OPPORTUNITIES

3.1 Background

In 1983, Buxton introduced a taxonomy of input devices that was focused on the human motor and sensory system [14]. He described how various transducers capture the human gesture appropriate for articulating particular intentions. Input devices were categorized by characteristics, such as the property sensed (position, motion, or pressure), the number of dimensions sensed, and the muscle groups required to use them. Research at Xerox PARC extended this work to capture a broader part of the design space of input devices [16]. The revised model captured both the discrete and continuous properties of devices.

In today's computing ecosystem, desktop and phone-based input interactions are not always appropriate for wearable scenarios. While the work at Xerox PARC captured the discrete and continuous properties of devices, I focus on the discrete and continuous properties of the interactions enabled by a set of one-handed input techniques that target the arms, hands, and fingers as input effectors. In this section, I highlight wearable interaction techniques that allow a range of inputs across a diversity of application scenarios and body parts. The work described here informs the projects detailed in this dissertation.

I began this dissertation quoting Buxton and pointing out that there are major shortcomings in our ability to manually enter information into a computer, in particular for mobility scenarios where a keyboard and mouse are no longer available. Input remains of critical importance. Guided by an exploration of the literature, I define the subset of the space of one-handed interaction techniques my work focuses on as follows:

- The interaction techniques support discrete input (e.g., taps, flicks, clicks), continuous input (e.g., cursor tracking, “drawing", analog values), or hybrid input as a com-

bination of discrete and continuous input in parallel (e.g., tapping while continuously moving the arm).

- The interaction techniques leverage parts of the body that provide fine dexterity or subtle input for manipulation and control, such as the fingers, arm, wrist, and mouth.
- The interaction techniques utilize devices worn on the body bringing the experience closer to the senses and making it quickly accessible for access when needed.
- The interaction techniques enable one-handed input, meaning that the use of the other hand is not necessarily required for their use. For example, tapping gestures of the thumb against a phalange or blowing at your wrist.
- Taking advantage of humans' kinesthetic sense allows for the input to be decoupled from the output feedback provided to the user. This enables more natural interactions with disaggregated devices on the body or without looking at the smartwatch.

3.2 Categories of Gestures Based on Intent and Usage

Gestures may be categorized based on their context of use and the intent within a user interface. These three categories are: activation, notification-response, and command. I briefly describe each of these below. Throughout the dissertation, I will refer to these types of gesture interfaces based on their intended purpose.

Activation — Activation gestures, as the name implies, are used to 'activate' some capability or capture the attention of the system (i.e., smartwatch or other device). The activation could be an indication of the user's intent to perform a command gesture or it could be used to wake up the system to perform a given task. Activation gestures may also be known as 'clutching' mechanisms to activate an interface before performing recognition. For example, speaking the words "Ok Google" or "Hey Siri", common on mobile devices today would be considered activation events prior to performing speech recognition. Ideally, activation gestures must be robust to noise and unique to avoid false positives.

Notification-Response — Notification-response gestures are performed in response to a notification from the system. The notification is meant to alert the user and give them the opportunity to 'respond' with an input event. The understanding is that the system is already listening for the response from the user. The response is only expected when the user has received a notification prompt. An activation gesture is not needed.

Command — Command gestures are used to provide explicit intent within the context of an application or interface. The action is typically attached directly to a corresponding action in the interface. In general, command gestures may also be more diverse and plentiful since they are used to select multiple actions within applications. Similar to touch gestures (e.g., pinch to zoom), every command gesture does not necessarily need to be unique but their use should be consistent across applications. An activation gesture can be used to both enter and end command mode, delimiting input actions.

A number of factors would influence which gesture is appropriate for each task or interface. Determining which gestures are most appropriate remains the task of interaction designers, developers, and device manufacturers.

3.3 One-Handed Input Taxonomy

I develop a taxonomy for one-handed input techniques based on a survey of related work in the space of gestural interfaces, including some of the work detailed in Chapter 2. The goal of the taxonomy is to provide a framework that can be used to categorize gestures, identify the diversity and breadth of gestures that have been explored, and understand how they can be incorporated as part of user interfaces. This summary also serves to identify gaps in the literature that inform the design of the systems developed in this dissertation.

The taxonomy spans related work in the literature across conferences and communities, including UIST, CHI, UbiComp, ISWC, MobileHCI, and others, with coverage over the past 5 years and some earlier work.

I focus primarily on categorizing based on the various body parts used to perform the

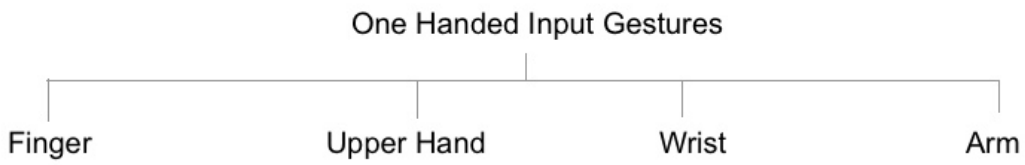


Figure 3.1: The top level of the taxonomy describing a broad classification of one-handed gestures based on the body part used to perform them.

gestures, not the sensing used to detect or pipeline used classify the gestures. A description of sensing techniques for input is presented in related work (see Chapter 2). Sensing of the gestures can be performed directly by instrumenting that body part or indirectly by instrumenting the environment or neighboring body parts.

In particular, I focus on the arm and its subparts. The arm consists of several segments that together make it one of the most useful and complex tools of the human body. These parts are: the upper arm, elbow, forearm, wrist, and hand. The upper arm extends from the shoulder to the elbow and provides pulling and lifting strength. The elbow is a hinged joint that allows the arm to open up to 180 degrees at full extension. The forearm is the area between the wrist and the elbow. The muscles in the forearm rotate, flex, and extend the wrist. The wrist is located in the upper hand. In total, eight carpal bones - along with multiple muscles and tendons - form this intricate area. Finally, the hand (palm and fingers) allows humans to do much more complicated tasks than any other animal, in particular because of its five fingers.

I narrow the segments of the arm for the purposes of this taxonomy down to four major body areas that are used to perform the gestures (see Figure 3.1). The four categories are: fingers, hands, wrist, and arm, in increasing order of coarseness of interaction granularity. I separate the hand to cover gestures that use both the palm and the fingers, and the fingers only. For the arm, I include the upper arm, elbow, and forearm. In the following sections, I describe the gestures identified in prior work and frame them within the context of this one-handed input taxonomy. I also present opportunities for future work.

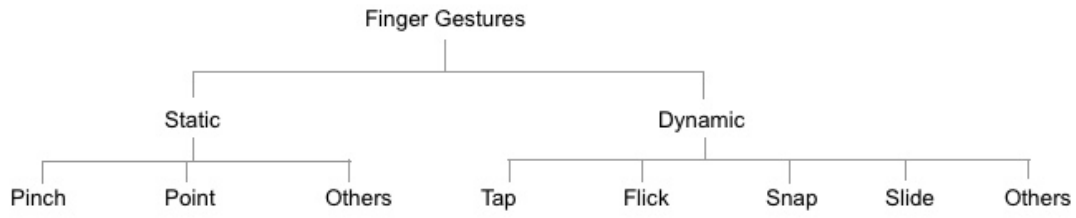


Figure 3.2: Types of finger based gestures, classified under static and dynamic type.

3.3.1 Finger Gestures

Gestures in this category are performed using one or more fingers of the same hand. Finger gestures are generally more discreet and much finer than hand- or arm-based gestures. Based on the survey of related work, I find that this category is particularly rich in gestures as compared to other categories, owing to the natural dexterity of the fingers and the large number of configurations they support. This rich set of gestures can be classified into static poses and dynamic gestures (see Figure 3.2). Static poses are those events that only require movement to assume the position that is detected by the sensory hardware. Dynamic gestures, as the name suggests, are detected based on motion of the gesture itself. I present a breakdown of finger gestures and classify them as either static or dynamic.

Static Finger Poses

The subset of static finger poses include finger pinches, counting, and other configurations of the fingers.

Pinching — One recurring and commonly found pose is the finger *pinch* [22, 29, 58, 60, 66, 99, 114, 123]. The pinch is typically performed using the thumb and any one of the other four fingers on the same hand, involving two fingers at any point in time (see

Figure 3.3). The gestures are primarily detected indirectly by sensing changes in contour or acoustic signals at the wrist, or electrical signals at the forearm. In some cases, it is possible to instrument one or more of the thumb or fingers.

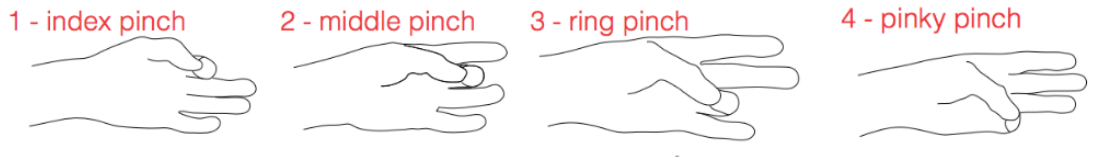


Figure 3.3: Pinch gesture performed between the thumb and each of the other fingers. The dynamic version of each of these pinches are detected as taps.

Counting — Another common group of static finger poses are *finger counts* [29, 60, 66, 81]. Counting gestures are most heavily represented in American Sign Language (ASL) and involve holding up one or more fingers to represent a number, while holding the other unused fingers down and rolled against the palm. See Figure 3.4.

Pointing — Similar to counting finger poses, individual fingers can be used as pointing mechanisms [60].

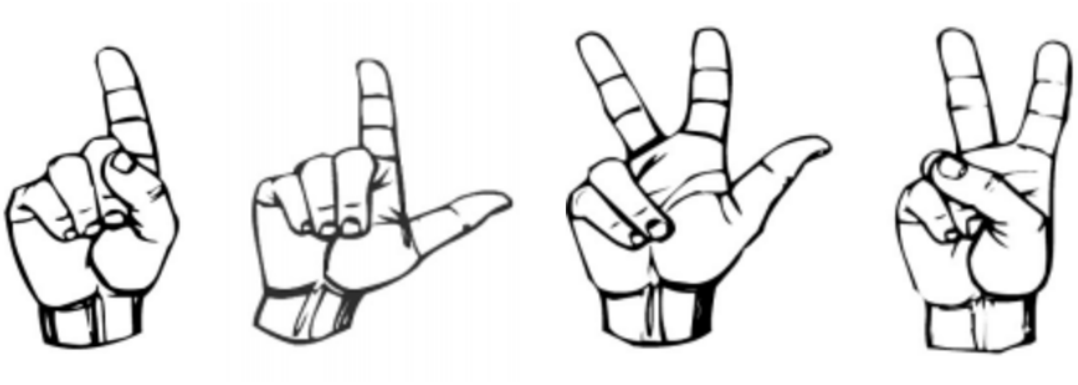


Figure 3.4: Some pointing/counting static poses performed by holding up a set of fingers.

Other Finger Configurations — WristCam [111] used a small camera mounted on the under-side of the wrist and aimed at the palm and fingers of the hand. It describes a new space of 3-finger gestures based on lifting combinations of one or more of the three middle

fingers (ring, middle and index) backwards or upwards from a rest position, resulting in 8 total detectable poses. Digits [60] also used a camera on the under-side of the wrist and identifies key parts of the hand, such as the tips and lower regions of each finger. Using a kinematic model, the system detects various poses, including the *Spiderman* pose, where the thumb pinches the middle and ring fingers together, while the index and pinky are pointed and stretched out. Some ASL gestures are also recognized, as documented in Backhand [66]. Some of the gestures in this work may also overlap with pinch, counting and pointing gestures.

It is possible for static finger poses to lend themselves to parametrization. For example, the duration of a pose could serve as a parameter, i.e., you hold a thumbs-up and the volume increases as long as the user maintains the pose.

Dynamic Finger Gestures

The subset of dynamic finger gestures include finger taps, flicks, snaps, sliding, and other combinations.

Tapping — A tap is similar to the pinch gesture described earlier. However, the main difference is that a tap is a pinch followed by a quick release of the pinch action and returning the fingers to a neutral rest state, as opposed to maintaining a pose. A number of related papers make use of tap gestures for input [2, 42, 52, 58, 109].

Taps are typically performed between the thumb and any one other of the four remaining fingers on the same hand. Additionally, taps can also be performed by tapping the thumb to any of the phalanges, the flesh areas between the knuckles, on any of the 4 fingers, as is demonstrated in ThumbRing [109], DigitSpace [52], and Thumby in this dissertation. There are a total of 12 phalanges that can be tapped by the thumb, three on each of the other 4 fingers, making that a rich total of 12 possible gestures. ThumbRing presented empirical evidence noting it is more comfortable for users to tap the side of the phalanges, instead of the palm-facing front side. Another slightly different version of the tap only involves

one finger, where the user taps his finger onto an imaginary surface, in-air, as described by Kerber et al. [58]. Taps are very versatile; in that they can be performed multiple times in quick succession. For example, single taps, double tap [2, 58] or possibly even triple taps, can be counted as different gestures. These combinations of taps may increase the richness of interaction.

Flicking — There are multiple ways to perform a flick gesture depending on the finger that serves to 'catch' or restrict the flick movement. Skinput [42] detected flicks of the finger using the thumb as a catch. Amento et al. [2] also recognized a flick gesture using a wrist-mounted bio-acoustic interface.

Snapping — Snapping the fingers is a well-known gesture that involves a 'rub' of the fingers and also produces some auditory output. Similar to the detection of the *flick* gesture, Amento et al. [2] also used a wrist-mounted piezoelectric microphone to detect snaps as input using a bio-acoustic system.

Sliding — Slide gestures are performed by sliding the thumb (typically) along other fingers or the palm of the hand (see Figure 3.5). The other fingers and palm become a canvas for the thumb to 'draw on'. Gestures can be designed based on the pattern the thumb follows across the surface of the hand. The patterns may be unique and known shapes such as a circle, as in PinchWatch [67], or they could be in the form of letters, as in DigitSpace [52]. DigitSpace also explored using the thumb to write graffiti letters on the surface formed by the other fingers of the same hand.

Alternatively, slide gestures may also be performed without having to draw a shape or pattern. ThumbSlide [3] used the whole length of the thumb to slide along the side of the index finger, while the hand is held in a fist. The thumb can also be used to slide along the length of individual finger segments, as also demonstrated by DigitSpace and PinchWatch. The direction of the slide provides an additional dimension to this kind of input. A slightly less obvious sliding gesture is the finger rub, demonstrated in Serendipity [114] and by Amento et al. [2], where the thumb slides against the other fingers repeatedly in quick

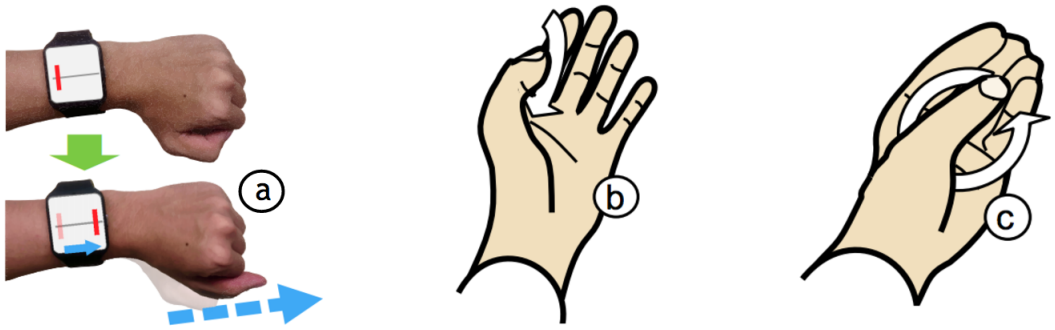


Figure 3.5: (a) Slide gesture used in Thumbslide [3] (b) Slide along individual finger segments (c) Sliding the thumb on the palm and fingers as a canvas to draw specific shapes, in this case, a circle.

succession.

Freeform — Apart from the categories mentioned above, some prior work also describes freeform continuous finger movements and tracking as input gestures. uTrack [18] by Chen et al. demonstrated continuous 3D input using a thumb-mounted magnet and two magnetometers attached to the fingers. The thumb moves freely and is tracked continuously in 3D-space. iRing [81] also demonstrated how bending of a finger can be measured when the ring device is worn on that finger. In these cases, the path or angle of a movement are tracked and mapped to an input which is useful for cursor and valuator controls in graphical user interfaces.

Similar to static finger poses, discrete finger gestures involving dynamic movements may also be parametrized. As mentioned earlier, the number of taps or repetition of other gestures might be detected and either used as as an input parameter or classified as different gestures.

3.3.2 Hand Gestures

I categorize hand gestures as all gestures that involve a combination of the entire palm of the hand and fingers used together. This class of gestures does not provide fine grain granularity and the same dexterity that finger gestures offer. Given the coarse granularity, there are fewer gestures and prior work in this category.

There is some ambiguity between distinguishing hand gestures and finger gestures. For the purposes of this taxonomy, I assume hand gestures engage all the parts of the hand between the wrist and the finger tips, together as a whole. All the fingers work in parallel to form an upper hand gesture, so no individual finger movements are part of the gesture as they were in finger gestures. Similar to finger gestures, I can classify hand gestures as static or dynamic.

Among static hand poses, the two most common gestures are the *making a fist* [3, 29, 58, 66, 68, 114, 123] and *"halt" or "stop"* hand gestures [22, 29, 58, 60, 68, 111, 123].

Dementyev et al. [22] and Zhang et al. [123] used pressure sensing and impedance tomography to detect a relaxed hand position, which is a neutral position with the palm open and in which the fingers are not forced into a spread-out position. Digits by Kim et al. vision-based [60] and Saponas et al.'s electromyography [99] systems also recognized a *grasp* gesture for both small and large objects independently (see Figure 3.6). Additionally, both papers described the *hook* gesture that is used while picking up a bag by its handles. Static hand gestures generally describe different degrees of opening or closing the grip of the palm and fingers.

There are few papers for dynamic gestures using the entire hand without actuation of the individual fingers. There is potential to explore new gestures within this category. One example of dynamic hand gestures is presented in Serendipity [114]. Their wave gesture involves bringing all 4 fingers (excluding the thumb) to a fist and rapidly returning to a resting state.

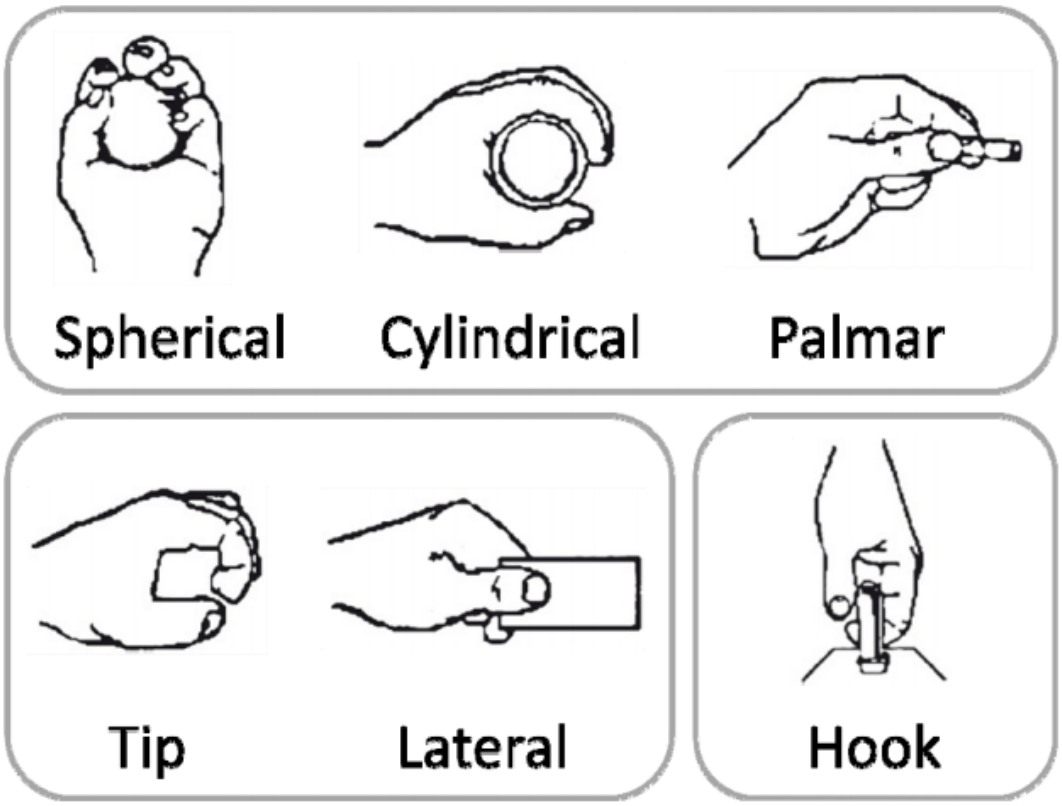


Figure 3.6: Set of the kinds of hand grips detected by Saponas et al. [99].

3.3.3 Wrist Gestures

Wrist gestures are those performed by bending the wrist joint. For example, as a flexion or extension of the wrist joint with fingers extended (see Figure 3.7). This has been explored by Kerber et. al. [58], Zhang et al. [123], and Lu et al. [68] with various sensing approaches (electromyography, impedance tomography, and a combination of electromyography and accelerometry). These gestures have also been explored commercially in the Thalmic Labs' Myo, sensed using forearm electromyography. While all of these papers discuss detecting the same movement, there are slight variations to how the gestures are performed. For example, Lu et al. used a closed fist while flexing or extending the wrist, while in the other two papers the fingers are straight and palm is open. Based on how they are performed, wrist gestures may be either static or dynamic. If the gesture entails repeated flexion,

extension, or both to be detected, I consider it a dynamic gesture. However, if the gesture is recognized by simply performing a single flexion or extension, and the wrist is held at that position or pose, I consider the gesture to be static similar to finger poses.

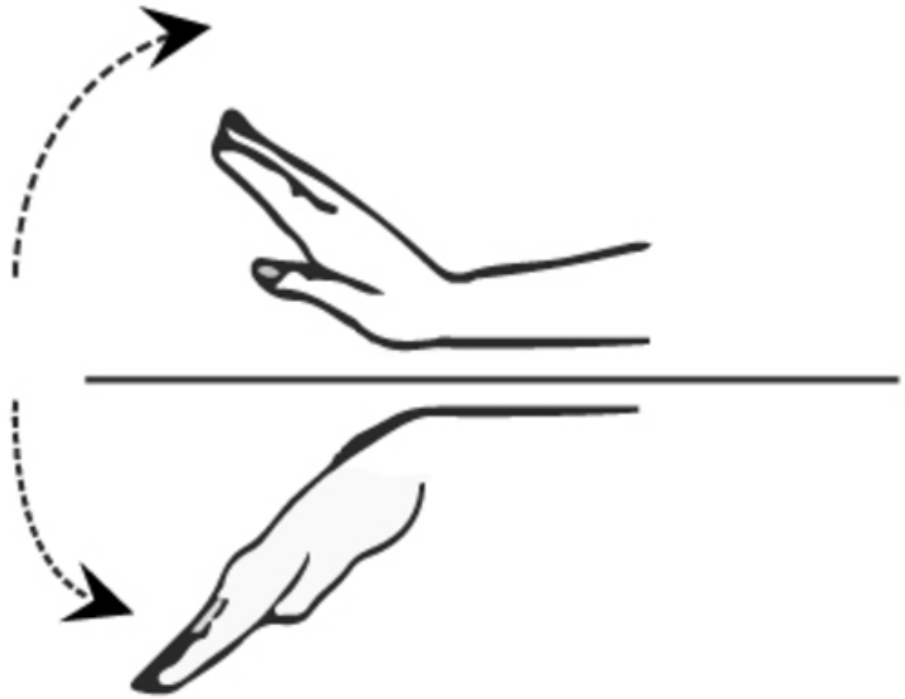


Figure 3.7: Wrist flexion and extension.

3.3.4 Arm Gestures

Arm gestures are those that are performed using the whole forearm, from the elbow up to the fingertips. These gestures result in a gross movement and comprise a smaller set of gestures, compared to the wrist, hand, and fingers. Similar to wrist movements discussed earlier, all arm gestures can be classified as either static or dynamic.

Freeform

Freeform arm gestures are performed by gross movement of the forearm around all three axes, while pivoted at the elbow. For example, these may constitute an up/down motion,

left/right motion, as well as a twisting motion of the forearm along its own axis. Lu et al. [68] recognized gestures along two of these axes — up/down and left/right. Additionally, this work also combines movement along these two axes to identify movement of the forearm in a circular motion along a plane perpendicular to the length of the forearm. See Figure 3.8.

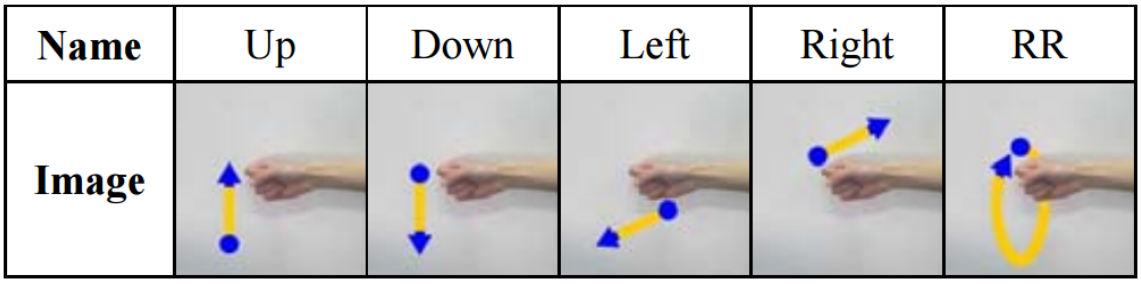


Figure 3.8: Examples of freeform large scale arm gestures as recognized by Lu et al. [68].

Tilt

Another gesture that has been used as an input modality, in particular for smartwatch input, is arm tilt. Combining all degrees of freedom of the arm enables greater control, as compared to the freeform gestures discussed in the previous section. In general, tilt gestures are used to interact with objects on a smartwatch screen worn on the same hand. The gestures are performed using subtle tilting movements along all 3 axes anchored around the watch screen. Another observation is that in general the gesture is intended to be performed when the arm is raised in a position where the watch screen is in the field of view of the user. On tilting, the direction of tilt towards gravity is measured. Guo et al. [35] used tilt gestures to select objects on the periphery of a watch screen, as well as to control movement of an object on the screen as it would naturally move due to the tilt direction and gravity. See Figure 3.9 for a visual representation of the interaction. In WristWhirl [31], wrist motion was used to continuously control a 2D cursor, while a thumb-to-finger tap is used for selection. InclineType [33] also used tilt to select letters on a watch screen for text input.

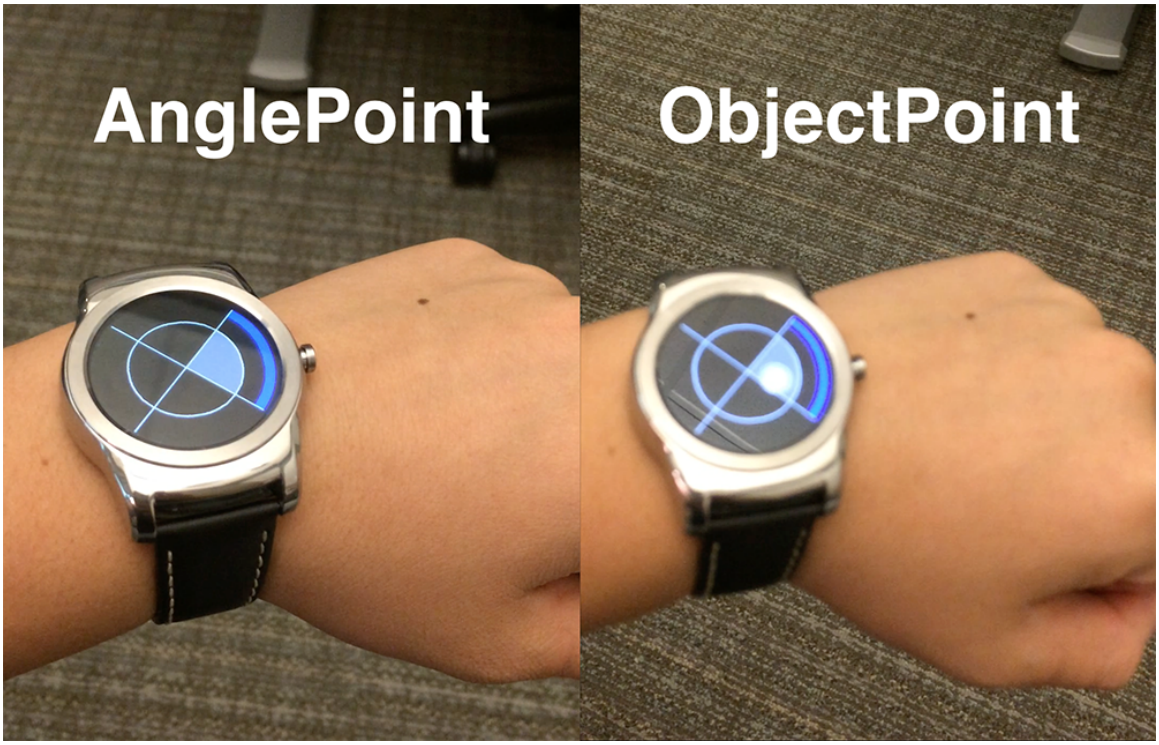


Figure 3.9: Arm tilt captured within an application using ObjectPoint and AnglePoint as seen in work of Guo et al. [35].

3.4 Dealing with Ambiguity in Categorizing Gestures

In some cases, there may be ambiguity between whether a gesture is static or dynamic. If a gesture requires motion only to assume a given pose, after which the pose itself is held static while it is recognized by the hardware, I consider it to be a static gesture. On the other hand, if the gesture is detected, not by a pose, but a series of positions or a specific motion, I consider it to be dynamic. The same gesture, however, can be interpreted either as static or dynamic, based on the intent of interaction and the sensing hardware. Take for example an arm tilt gesture that is used to control the movement of an object or cursor on the screen, as demonstrated in the work by Guo et al. [35] where the researchers describe two applications of the tilt gesture. The first is *AnglePoint*, where on tilting the arm and holding it at a particular angle in the tilted direction, the user is able to select an object along the rim of the watch screen. This can be interpreted as a sort of static gesture, because motion

is only used to assume the tilt direction, after which the arm can be held steady. However, in the second application *ObjectPoint*, the tilt of the watch is used to move a cursor in real time on a watch screen. This is a more responsive dynamic motion than the singular tilt in *AnglePoint*. This makes it hard to exactly assign a static or dynamic metric to some gestures unless additional details and understand of the usage context are available. It is up to the interaction designers and developers to determine how gestures will be used within an interface and how the capabilities and properties of that gesture are communicated to the user.

3.5 Opportunities Identified from One-Handed Input Taxonomy

In the previous sections, I discussed a number of interaction techniques and a taxonomy primarily focused on the use of the arm, hands, and fingers. The systems described in the taxonomy primarily rely on isolated detection of events or gestures over time. I have identified and developed three opportunities to enhance the input taxonomy with three new concepts of interaction. The first opportunity is to broaden the set of one-handed gestures supported by a single system. The first interaction technique in this dissertation focuses on detecting a broad set of one-handed gestures using multiple body parts (wrist and thumb) and sensing them with minimal instrumentation. The second opportunity is the introduction of rhythmic repetition between the interface and user's gestures. The second interaction technique introduces synchronous gestures into the space of wrist-worn devices by detecting rhythmic and synchronous movements of the wrist and fingers to a given on-screen stimulus. Other synchronous gestures have primarily been focused on eye-tracking and computer vision approaches. The third opportunity is to extend the taxonomy and incorporate other body parts not traditionally used for actuation. The third interaction technique in this dissertation uses the mouth as an actuator for non-voice acoustics captured by the smartwatch, extending work in blowable interfaces on the desktop and other non-linguistic vocalization systems.

3.5.1 One-Handed Gestures Using Multiple Body Parts and Minimal Instrumentation

While developing the presented taxonomy, I observed that the majority of papers were focused on a narrow subset of gestures with a single body part. In most cases, the limitations were imposed based on hardware constraints or predetermined interaction goals. I see this is as an opportunity to develop systems that span a broad range of gestures using single- or multi-point sensing by instrumenting individual or multiple body parts.

So far, I have described distinct categories for gestures of different granularity and uniquely tied to distinct body parts. It is also possible to combine (atomic) gestures from two or more of the above body parts to form a combination of atomic gestures, or compound gestures, that fall under a hybrid category of multiple body parts. For example, Kerber et al. [58] combined a pinch gesture with arm rotation along its length, thus forming a compound gesture. Another example of a compound gesture is tapping the thumb-to-index finger at different arm orientations. Although Lu et al. [68] detected gross arm movements, it required the user to hold their hand in a fist while performing them.

There are various reasons why one might want to use compound gestures. One reason is to reduce false positives, or unintended input that a wearable device can detect, if gross movements are performed. A fine finger pose held over a brief period of time can act as an activation gesture or a way to inform the system to listen for gross gestures. Alternatively, a gross gesture of the arm can provide context to a finer finger gesture. For example, performing a tap when the arm is resting by a user's side could be interpreted differently from the same tap gesture with the arm in a raised position. Alternatively, the gesture performed when the arm is raised and the arm is at rest could be interpreted as the same gesture as a system that is independent of rotation. Thus, compound gestures designed by combining two simple gestures or by instrumenting multiple body parts can add to the richness and dimensionality of input actions.

In Chapter 4, I introduce an interaction technique called Thumby. This technique is a gloveless, inertial-based technique that combines sensors mounted at the wrist and on

the thumb to sense thumb and wrist movements and enable a broad set of finger-level gestures. I demonstrate the ability to perform multiple finger-level gestures and present a system that is rotation-invariant to arm placement and device placement. This work seeks to broaden the possibilities of gestural interfaces using minimal instrumentation on the wrist and fingers and hope it encourages further innovation in the design of gestures available to everyday computing users.

3.5.2 Synchronous Gestures for Wrist-Worn Devices

The second opportunity is the introduction of synchrony in response to a given user stimuli.

Hinckley et al. [47] proposed synchronous gestures as a new interaction metaphor for distributed sensing systems, highlighting their technique using two tablet devices. This early work characterized synchronous gestures as single events co-occurring in time across multiple devices or people. For example, bumping two tablets together or two people wearing an accelerometer-augmented watch to sense shaking hands. In the tablet bumping scenario, each tablet experiences a roughly equal but opposite pattern of forces. Bumping the left side of a tablet versus striking the right side results in a similar sensor pattern, but with spikes in the opposite direction. This allows the system software to determine which side of each tablet has made contact. Hudson et al. [53] presented an inattentive interaction technique, i.e., users can interact with devices without much attention, that relies on detecting multiple "whacking" events over time. Different from Hinckley et al.'s work, this approach looks at repetition over time with a pattern that is previously known to the user (e.g., whack, whack, whack) on a single mobile device, as opposed to across multiple devices. Recognition in both of these techniques depends on known patterns of sensor data for a particular gesture. More importantly, the user does not receive any stimulus on how or when to perform the gesture.

On the other hand, recent work has demonstrated techniques that leverage spatial synchrony for controls where the user gestures in response to a specific stimulus provided to

them. In particular, projects have focused on large public display interactions. Pursuits by Vidal et al. [112] demonstrated an eye tracking method that, instead of using the direct eye gaze coordinates, is based on the trajectory of raw eye movement compared to the trajectories of objects in the field of view. PathSync by Carter et al. [17] extended the idea of matching trajectories to hand movements, a principle the researchers refer to as rhythmic path mimicry, captured using a depth camera in front of a public display. While previous techniques relied on a single modality using particular body parts (e.g., eyes or hands), TraceMatch [21] used a conventional webcam for movement correlation enabling users to produce matching gestures with any given body parts. These techniques offer intuitive, walk-up interfaces and single- and multi-user input with various body parts. However, they depend on tracking the user while they are stationary and are not specifically designed for everyday wearable computing scenarios.

Eye tracking techniques can be extended for use during everyday activities with the availability of mobile eye trackers. Orbits by Esteves et al. [25] enabled hands-free input on smartwatches by using an external eye tracking device to match the smooth pursuit movements of the eyes to the path of a target on the smartwatch screen. Their technique detects whether the user is looking and at which target. Some of the major disadvantages of eye tracking are that our eyes are used for daily activities and navigation leading to inevitable distractions, sensing is susceptible to ambient interference, and vision-based eye trackers typically require significant power and computation. Dhuliawala et al. [24] further extended the use of smooth pursuit movements, indirectly tracking eye movement captured with a pair of electrooculography (EOG) glasses. EOG glasses account for some of the limitations of vision-based eye tracking devices by eliminating ambient interference and high computation cost. However, as discussed in their paper, a limitation of EOG is that significant noise was observed in the signals when the participant moved their head. Thus, requiring the user to be stationary while tracking an orbit.

In Chapter 5, I present SynchroWatch which is, as far as I am aware, the first system to

use sensors in a commodity smartwatch and rhythmic movements for input. The system depends only on a passive magnetic ring without the additional expense, bulkiness, or power requirements of custom solutions. For example, Orbits and Dhuliawala et al.'s work also used synchrony for selection in a wearable interface using eye-tracking and EOG glasses. However, the SynchroWatch technique does not require any additional hardware beyond the smartwatch and a small passive magnetic ring. Additionally, the interaction is designed to be robust to global motion (i.e., walking) enabling quick input on-the-go, while the majority of other synchronous interfaces have only focused on stationary scenarios.

To summarize, SynchroWatch focuses on repetition of a particular thumb gesture and measures the relative change in the signal over time. I describe the implementation of synchronous gestures for smartwatch interactions and suggest avenues for further development in this space.

3.5.3 Non-Voice Acoustic User Interfaces

The third opportunity I have identified is to extend the interfaces in the taxonomy to other body parts (e.g., mouth, legs, head, etc.) for actuation. One example of this type of interface is to use the mouth as an actuator.

Acoustic user interfaces can be characterized as speech and non-speech (or also known as non-voice). I briefly discuss speech and present more details on non-voice acoustics. Speech is a high bandwidth mode of communication that people use everyday to interact with others. In computing, the most common use of speech is to convey intent to a computer, mobile phone, or most recently a wearable device. For desktops, speech is primarily used for commands to applications, dictation to enter text, or controlling a cursor by commanding phrases.

Speech offers an expressive alternative to on-screen input, but non-speech acoustics may also provide a secondary input channel. Various types of non-speech input such as humming and whistling have been used to provide continuous input [102], and Igarashi et

al. demonstrated how duration, pitch and tonguing of sounds can be used for interactive controls [55]. Closely related, others presented interaction techniques using prosodic features of speech and non-verbal metrics [37, 82]. Sakamoto et al. proposed a technique to augment touch interactions on a mobile device with non-voice sounds as a parallel input modality [97].

Blowing is another type of non-voice acoustic interaction in the literature, used for selection, gaming, entertainment, accessibility, or text entry [28]. Zyxio's SensaWaft used a MEMS-based sensor array in a headworn microphone to detect blowing, enabling bidirectional controls for scrolling, zooming, and rotating a button dial [113]. BLUI is a fingerprinting technique that localized where a person is blowing on a laptop screen and demonstrated accuracy of over 95% for 4x4 regions with a single microphone [86]. Chapter 6 is inspired by BLUI's initial results, and Whoosh seeks to not only localize blowing on a different form factor with a significantly smaller screen but also to capture how a person is acoustically interacting with the device. Blowatch proposed blowing air at a smartphone prototype with four external microphones simulating smartwatch interactions [19], and presented a taxonomy for blowing events on a watch.

In this dissertation, I present a system called Whoosh (see Chapter 6). I extend the taxonomy presented by Blowatch with a personalized event set which includes blows and other types of non-voice acoustic input detected by the microphone on a commodity smartwatch device. The system uses machine learning to distinguish different types of acoustic events (e.g., blowing, shushing, sipping, puffing, etc.). I also introduce passive instrumentation using a 3D-printed watch case that increases the expressivity of the input set.

In summary, Whoosh extends the use of non-voice acoustics to wrist-worn devices and provides a fast, subtle, and expressive means of interacting with the smartwatch on-the-go. I present the implementation of the system and opportunities for future work.

3.6 Summary

I present a survey of one-handed interaction techniques and a taxonomy consisting of arm, wrist, hand, and finger gestures. The work in this chapter serves to identify gaps in the literature that inform the design of the systems developed in this dissertation. The majority of prior work covered in this taxonomy build specific sensing techniques to detect gestures that fall under one of the branches of the taxonomy (i.e., arm or finger or hand or wrist), and this was often due to the nature of the sensors or wearable device conceived to detect a particular kind of input event. Typically, this approach is characterized by a technology-first view of building interaction techniques. In particular, throughout this dissertation I pursue an interaction-first approach that later informs the choice of sensing technologies. I also identify three areas of opportunity for further research. The first opportunity is to design systems that span a broad range of gestures using single or multiple body parts and minimal instrumentation (Chapter 4). Two areas that have been underexplored in the space of one-handed input include the use of synchrony between the interface and the user, as well as the use of non-traditional body parts for actuation. I present two interaction techniques to address these opportunities focused on synchronous gestures using magnetic sensing (Chapter 5) and non-voice acoustics for wrist-based systems (Chapter 6). The remaining chapters in this dissertation seek to fill the gaps in these three areas.

CHAPTER 4

THUMBY: ONE-HANDED THUMB INTERACTIONS USING WEARABLE INERTIAL SENSING

4.1 Abstract

Thumby is an interaction technique that uses inertial sensors embedded in a thumb-worn ring to provide one-handed, quickly accessible, gloveless, and eyes-free interaction capabilities. Thumby is designed to detect one-handed thumb-based gestures including pinch, tap, swipe, flick, and several others. I present two hardware solutions. The first solution, Thumby v1, uses a single standalone 6 degree of freedom (DoF) inertial measurement unit (IMU) on a thumb-worn ring. The ring is wired to a separate laptop and the system captures and analyzes the orientation and movement of the thumb. I describe the gloveless hardware and software solution which is capable of recognizing over 20 unique interactions using the thumb, fingers, and palm of the hand, and operates in real time. Results of two user studies with 14 total participants validate the technique as a viable input modality for microinteractions on smartwatches, head-mounted displays, and other connected devices. Thumby v2, a second version of the system, combines a 9 DoF wireless and battery-powered inertial measurement unit in a thumb-worn ring with a 9 DoF suite of sensors embedded in a smartwatch. The sensor data from the ring is captured via Bluetooth along with local sensor data by the smartwatch for offline analysis. A user study is conducted with a total of 8 participants performing 8 gestures, while sitting and walking. Finally, I address some of the limitations of the approach and demonstrate a variety of input scenarios for mobile users with a disaggregated set of on-body and surrounding devices.

4.2 Introduction

Technology advancements in computation and sensing have made it possible to carry devices small enough to fit in our pockets or wear on our bodies. Most of the activities supported by mobile computing rely on touch-based input on touchscreens. However, wearable devices such as smart watches or head-up displays suffer from tiny screens for output and limited surface areas for input, significantly hindering their interaction capabilities. Many of today's connected devices could potentially benefit from an input modality that is quickly accessible, available when needed, and that provides a diverse input vocabulary for discrete and continuous controls. Speech input provides a viable solution but is sensitive to ambient noise, not ideal for use in public spaces, and tedious for repetitive short microinteractions.

Alternatively, hands and fingers are ideally suited to interact with connected devices worn or carried on the body in a subtle, fluid fashion while on-the-go. The physiology and dexterity of the hand enables a wide variety of gestures, from pointing and rubbing to flicking and swiping, as well as tapping visual controls on touchscreens. Humans' unique sense of kinesthesia—the ability to sense the relative position (proprioception) and spatial orientation (vestibular system) of the body—allows us to perform finger gestures without visual attention. For example, pinching the thumb against parts of the fingers has been used for counting and may be performed without looking.

I introduce *Thumby*, an inertial sensing interaction technique that can recognize taps, swipes, flick, gestures, and unique thumb pinches against the phalanges (i.e., the flesh area between the finger joints). Two proof-of-concept thumb ring systems enable over 30 distinct input events to be detected without reducing the dexterity of the hands.

This work makes the following contributions to the input and interaction communities:

1. I describe the design of Thumby v1 — a 6 DoF wearable thumb ring system that detects one-handed thumb and finger interactions using inertial sensing

2. I provide empirical evidence of the performance and limitations of Thumby v1 through

two user studies with a total of 14 participants

3. Building on the initial design of Thumby v1, I describe the design of a second iteration Thumby v2 — a Bluetooth-enabled 9 DoF thumb ring and a smartwatch

4. I provide empirical evidence of the performance of Thumby v2 through a user study with 8 participants

5. I demonstrate the use of Thumby across multiple disaggregated mobile and wearable devices in meaningful input scenarios

4.3 Motivation

The thumb is essential for grasping and manipulating objects. The thumb has three joints (CMC, MCP, IP) that provide a total of 5 degrees of freedom, allowing a person to combine flexion and extension, abduction and adduction, opposition and reposition. See the taxonomy in Chapter 3 for additional details.

The initial motivation for this exploration was to capture the tapping of the thumb against the phalanges of the fingers on the user's same hand, which affords a large number of distinctly recognizable gestures. The phalanges are the bone areas of the fingers separated by creases. There are three phalange bones per finger and two for each thumb. The three phalanges in the fingers (from top to bottom) are named distal, medial, and proximal, each of which is covered by a layer of skin flesh and are the target of the pinch gestures.

I take advantage of the dexterity of the thumb to recognize a large event set to cover a broad range of application needs for one-handed wearable computing. In this chapter, the thumb is used for pinching, tapping, sliding, flicking, poses, and other finger gestures.

In the remainder of this chapter, I will describe two hardware explorations of the Thumby system. The first uses a standalone thumb ring with 6 DoF inertial sensing (accelerometer and gyroscope) wired to a neighboring laptop and is geared for interfacing with connected and neighboring devices (e.g., head-up display, projected screen, mobile

phone). The second solution uses a completely wireless and battery-powered 9 DoF inertial sensing (accelerometer, gyroscope, magnetometer) thumb ring. The solution is focused on smartwatch-based interactions. This version of the ring is connected over Bluetooth to a smartwatch with its own 9 DoF inertial sensing solution on Android. Thumby seeks to enable a broad set of finger/thumb gestures.

4.4 The Thumby System - Version 1: Standalone Thumb Ring

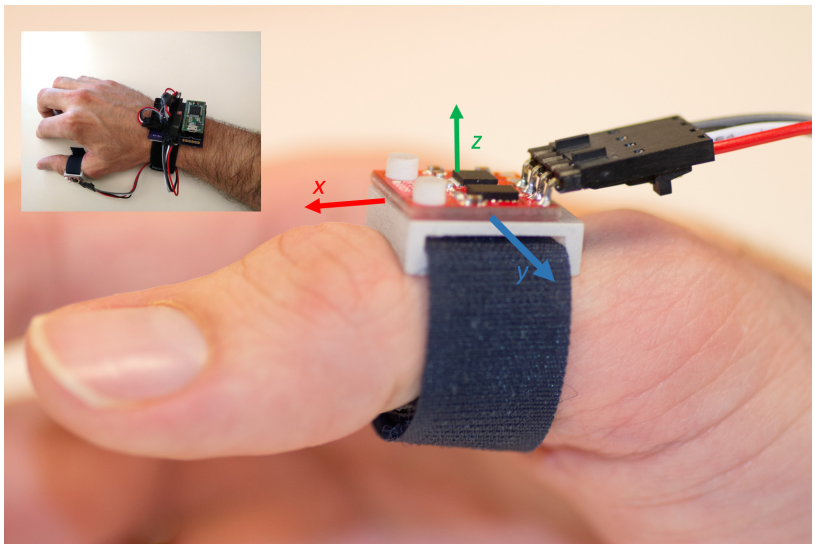


Figure 4.1: Thumby v1 is a wearable, gloveless, inertial sensing technique to detect thumb pinches, taps, swipes, flick and gestures using sensors at the thumb. Sensing elements detect the orientation and movement of the thumb.

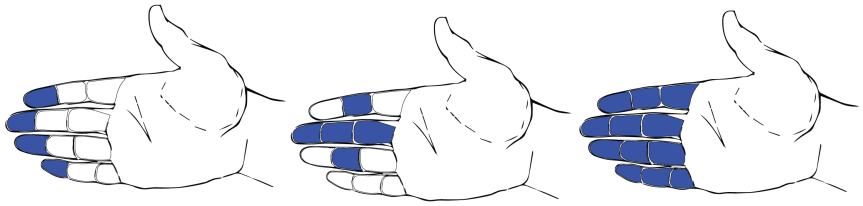
The first iteration of Thumby (see Figure 4.1) uses 6 DoF inertial sensors embedded in a thumb-worn ring to provide one-handed, quickly accessible, gloveless, and eyes-free interaction capabilities to connected devices. In this section, I describe the Thumby v1 system, including the supported interaction techniques, design criteria, hardware prototype, data pipeline, and technical evaluations.

4.4.1 Interaction Techniques

Motivated by the need for a wearable input device that provides a variety of natural interactions, the system takes advantage of the surface area of the palm of the hand and degrees of freedom of the thumb and fingers for input. See the taxonomy in Chapter 3 for additional details. The system provides the ability to detect thumb pinches, taps, swipes, and flicks. I abbreviate the set of thumb pinches using two digit codes. The first letter is the finger name (e.g., 'i' for index, 'm' for middle, 'r' for ring, 'l' for little) and the second digit corresponds to each of the phalanges (e.g. '1' for distal, '2' for medial, '3' for proximal) from top to bottom. Using this convention, for example, the fingertips are labeled 'i1', 'm1', 'r1', and 'l1. Other interactions are named semantically based on the action to perform (e.g., draw 'counterclockwise circle', 'swipe right', etc.).

Pinch

Pinches are useful for discrete inputs from a set of possibilities, such as 4-way button interfaces, directional controls, and alphanumerical input. I explore up to 12 distinct pinch events between the thumb and the phalanges of the hand, where the finger pose is maintained and the thumb remains stationary. The orientation of the thumb sensor relative to the ground is used to distinguish between different poses. I leverage the natural tesselated layout of the phalanges to explore three layouts for pinches: 4 phalanges using the fingertips



(a) 4 pinch condition (b) 5 pinch condition (c) 12 pinch condition

Figure 4.2: The thumb and fingers combined with inertial sensing in the Thumby system provide a rich input modality. Up to 12 phalanges are used as target locations for thumb pinches in three configurations (4 pinch, 5 pinch, and 12 pinch).

(i1, m1, r1, l1), 5 phalanges in a d-pad configuration centered around the middle finger (i2, m1, m2, m3, r2), and all 12 phalanges. I also include recognition for an open hand pose, for a total of up to 13 input events. The hand is placed in a relaxed state with the palmar surface perpendicular to the ground, similar to holding a cup. See Figure 4.2.

Taps

Taps and double taps provide an additional vocabulary of input symbols, and are useful for interactions such as binary state dialog boxes, selection, and shortcuts. Taps are rapid pinches of the thumb against the side of the index finger. Similar to mouse clicks, the system supports left tap at the index fingertip (i.e., i1) and right tap at the base of the index finger (i.e., i3). In this event, the thumb approaches from a relaxed state, impacts the finger, and retracts to open hand. The movement of the thumb is captured and its path is used to uniquely identify each event. The taps event set also includes double tap at the tip and base of the index finger – a total of 4 gestures. See Figure 4.3.

Swipe

Swipes are useful for indicating increasing or decreasing changes of value, such as flipping through arrangements of items (e.g., list items, pictures) or directional state changes (e.g., fast forward, previous song, or increasing/decreasing volume). The top lateral part of the flexed index finger is used as a surface area for thumb swipes. The index finger provides haptic feedback during the gesture, as well as guidance on range of thumb motion. I explore 4 distinct swiping gestures: swipe left-to-right, swipe right-to-left, multiswipe right-to-left, and multiswipe left-to-right. The direction of the movement is captured by the system. A swipe spans the side of the distal and proximal phalanges of the index finger (i.e., i1 to i3). The multiswipe event consists of three fast swipes in rapid succession. See Figure 4.4.

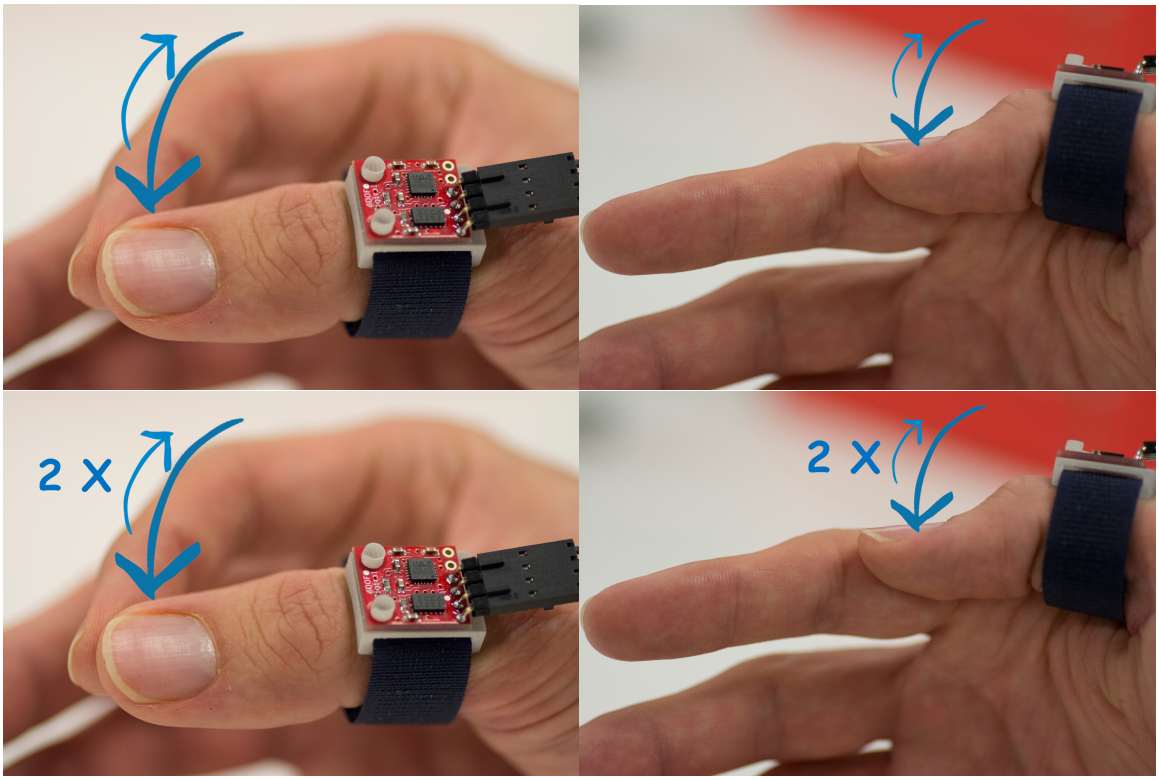


Figure 4.3: Examples of tap events include tap left and tap right. Double tap left and double tap right are uniquely recognized.

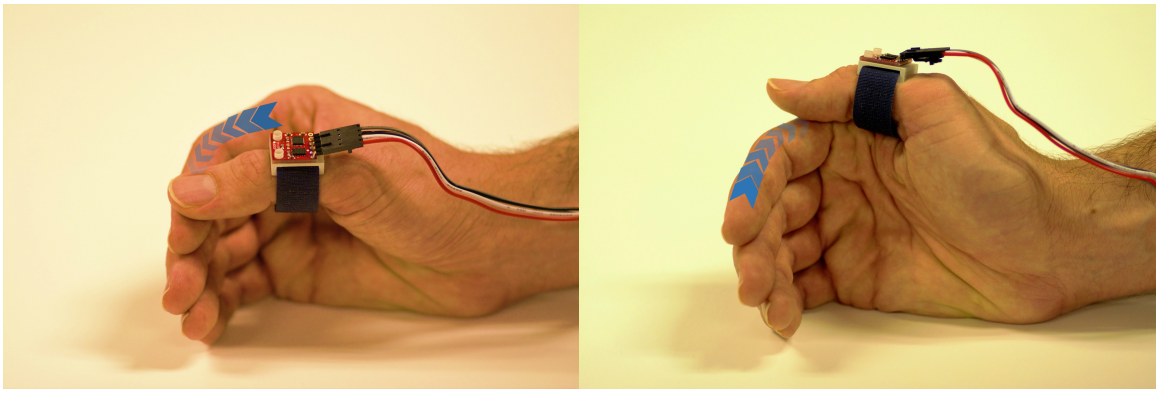


Figure 4.4: Swipe left and right. Multiswipes are three swipes in rapid succession (not shown in image).

Flick

Flicks could be used to dismiss notifications, pull up slider menus, or quick glance. The flick gesture is performed by pressing the nail of the thumb against the medial phalange of the index finger (i.e., i2), then releasing the thumb vertically while pushing away, and

returning to a relaxed hand state. I explored flick gestures at the three phalanges of the index finger but discovered confusion with tap events during pilot studies, thus focus the flick gesture at the center of the index finger. See Figure 4.5.

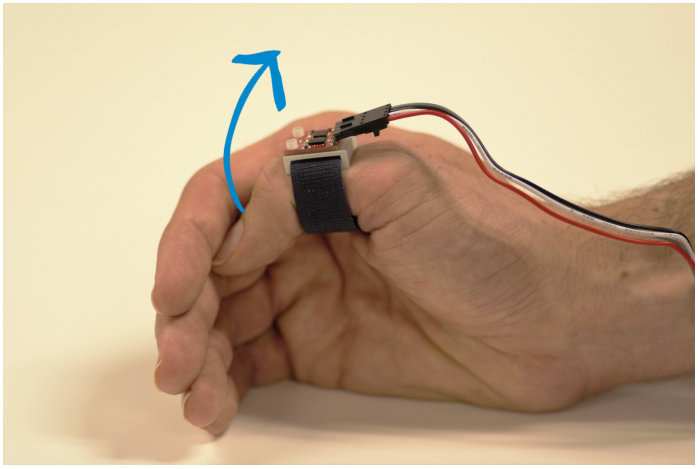


Figure 4.5: A flick gesture is shown

Drawing Gestures

Gestures are a natural modality for providing input and useful for sending commands to external devices (e.g., in a smarthome environment), mode switching between input modes, and mapping to shortcuts or actions within applications. I use the thumb to ‘draw" on the palm of the hand, with the entire hand resting at approximately a 45°with the palm front facing the user. I adopt the gesture set from Kela et al. [57] that includes 8 gestures - greater than (>), square, left-to-right, right-to-left, down-up, up-down, clockwise circle, and counterclockwise circle. See Figure 4.6.

4.4.2 Design Criteria

I summarize several criteria that influenced the choice of sensing technology and design of Thumby v1.

- *One-handed Operation*: The hands are one of the most dexterous parts of the human body and facilitate direct manipulation of objects and interfaces [48]. One-handed

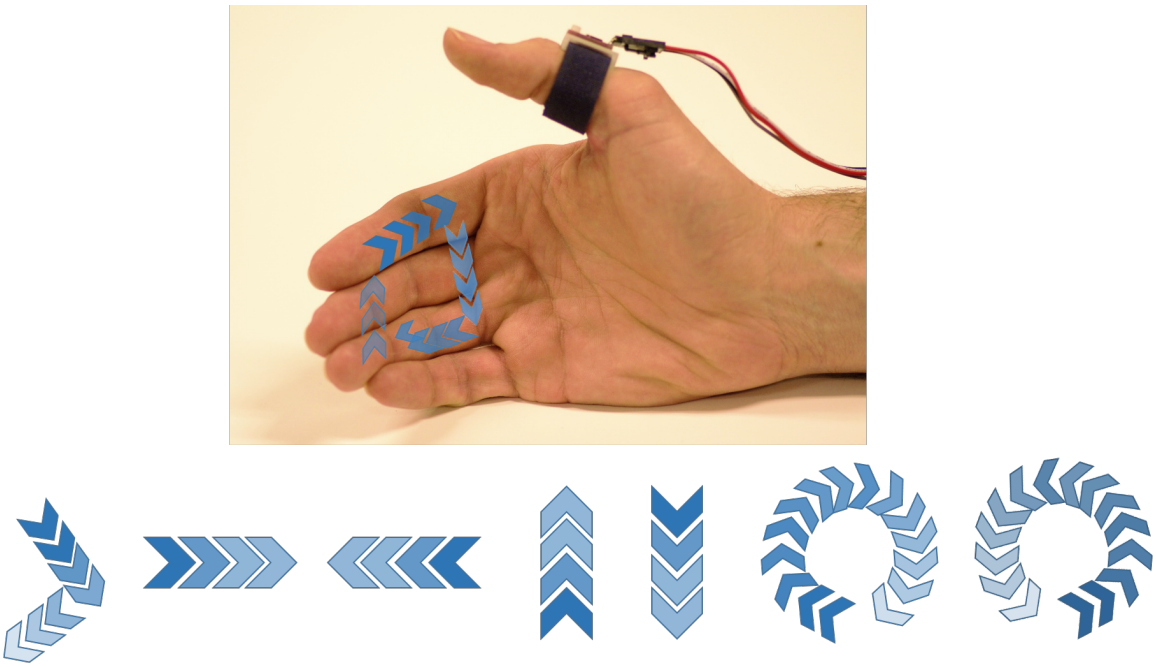


Figure 4.6: Examples of gestures

input controls can be directed to any device in the environment or on-body, while the secondary hand remains free.

- *Gloveless*: The the palm of the hand should not be instrumented with any materials or sensors that would be uncomfortable, cause loss of tactile sensation, or be socially awkward. A ring is widely considered to be a more socially appropriate form factor and does not affect the dexterity of the hand.

- *Eyes-Free*: The user should be able to provide input gestures using their proprioceptive abilities, without looking at the input device or their hands.

- *Quick Access*: Input needs to be always-available [76], and it should address the "2-second rule", in which a mobile user should be able perform an action within two seconds of forming an intent [105].

- *Disaggregated Input/Output*: The location of input gestures is decoupled from the output feedback location, enabling more natural interactions with disaggregated devices on the body or projection in the environment.

4.4.3 Hardware Prototype

The Thumby v1 prototype is a thumb-worn ring with a 3-axis accelerometer and 3-axis gyroscope (see Figure 4.7). The system is designed to capture movements of the thumb, fingers and wrist. The accelerometers measure the relative force due to gravity based on the orientation of the sensor relative to ground, while the gyroscopes measure angular rotation.

The thumb-worn IMU is wired to a custom printed circuit board (PCB) that sits on top of the wrist attached to a velcro band. The board includes a Teensy microcontroller for sensor calibration, data acquisition, on-board filtering, and transmission. A Bluetooth connection to a laptop for data collection is provided. The PCB also houses an I^2C multiplexer for interfaces with multiple sensors (if needed), battery connector, and outlets to sensor units. In the future, all components of the device could be designed to fit inside a single package worn around the thumb, similar to commercially available products such as MOTA SmartRing.[1]

The thumb-worn IMU is mounted on a 3D-printed pad with a concave curvature to sit comfortably on the thumb. The curvature also ensures that the device is centered on the top of the thumb, which is important for calibration and device usage over time (e.g., if the ring is removed daily).

The hardware specifications of the system include:

- 1 x 6 DoF IMU board. The IMU board provides an InvenSense ITG3200 3-axis digital gyroscope with a full-scale range of $\pm 2000°$/sec and an Analog Devices ADXL345 3-axis digital accelerometer with a range of ± 2g in a single convenient package. A second IMU is added to the system for later explorations.

- 1 x PCA9546A low voltage 4-channel I^2C multiplexer to interface with inertial sensors, expandable to four sensors total.

- 1 x PJRC Teensy 3.1 microcontroller. The Teensy is built around an ARM Cortex

[1] https://www.mota.com/doi-smart-ring/

M4 running at 72 MHz and is capable of high speed data acquisition and serial communication.

- 1 x Bluetooth RN-42 breakout for wireless sensor data transmission.

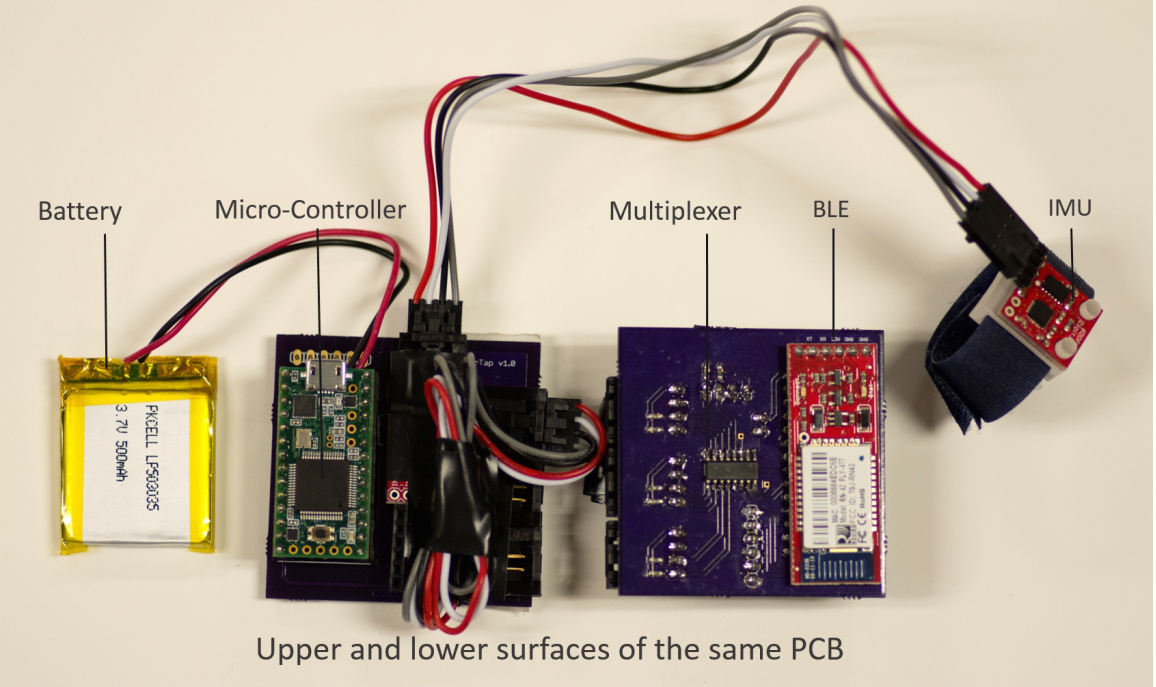


Figure 4.7: The Thumby system includes: 6 DoF IMU worn around the thumb, microcontroller, Bluetooth module, I^2C multiplexer, and battery.

4.4.4 Recognition Pipeline

In this section, I describe how the raw sensor data is turned into input events: sensor data acquisition, preprocessing and filtering, feature extraction, segmentation, and gesture classification.

Data Preprocessing and Filtering

The data captured by the 3-axis accelerometer corresponds to the superposition of acceleration experienced by the sensor mounted on the rigid thumb-ring pad and the impact of the earth's gravitational field. The data reported by each of the three perpendicular axes

is dependent on the orientation of the device towards ground. Orientation and position are crucial for detecting pinches and used for recognizing all interactions explored in this work. In addition to the accelerometer, the system uses a 3-axis gyroscope to capture the direction and deflection of thumb movement, useful in recognizing side to side swipes and more complex gestures. Gyroscopes are useful for measuring or maintaining orientation, due to the conservation of angular velocity, and are widely used in inertial navigation systems. The angular velocity is measured as zero when there is no movement, positive for rotation along the axis, and negative opposite the axis. I sample both the accelerometer and gyroscope at 100Hz, which provides sufficient data for classifying the interactions and using the system in real-time.

For preprocessing and filtering, I implement a low pass filter on the system's microcontroller using an exponential weighted moving average. The data is normalized and a complementary filter is used to stabilize the gravity vectors and orientation of the platform. The complementary filter compensates for gyroscope drift using the accelerometer, which is less prone to drift. Similarly, the gyroscope with high sensitivity and robustness to external forces is used to minimize accelerometer noise. For implementation details, I refer the reader to the literature [45].

Feature Extraction

I characterize the input events using 78 features for both training/testing frames of sensor data and build a model for real-time classification as well as input to the classifier. For taps, swipes, flicks, and gestures, I use a standard feature set [12, 115] that includes the mean, root mean square (RMS), standard deviation (StDev), average energy per sensor axis (AvgEnergy), and cross correlation for both the accelerometer and gyroscope (CrossCorr). Additionally, I extract the zero crossing rate (ZeroCross) for the gyroscope to capture directionality and repetition of movements (e.g., swipe and multi swipes left or right).

Finally, I use the empirical cumulative distribution function (ECDF) of the accelerom-

eter. The ECDF representation provides an objective measure to effectively capture and preserve the distribution of the accelerometer sensor data for each frame, typically lost in certain statistical features (e.g., mean). In particular, the spatial position and general shape of the representation are useful in distinguishing finger pinches and other events from each other. For pinch detection, I use only a subset of the feature set above that includes the mean of accelerometer axes as a measure of the sensor's relative orientation towards the ground and the ECDF. The ECDF provides additional robustness to slight movements of the thumb during a pose. I refer to Hammerla's work for a more detailed description and implementation of the ECDF [36]. The feature set is summarized in Table 4.1.

	Accelerometer (3-axis)	Gyroscope (3-axis)
Mean	3 features	3 features
RMS	3 features	3 features
StDev	3 features	3 features
AvgEnergy	1 feature x axis	1 feature x axis
CrossCorr	1 feature x axis pair	1 feature x axis pair
ECDF	15 features per axis	—
ZeroCross	—	3 features

Table 4.1: Summary of features extracted for recognition of taps, swipes, flicks, and gestures. A subset of these features (i.e., accelerometer Mean and ECDF) is used for detecting pinches.

Segmentation

The pinch segmentation uses a sliding window approach with standard 50% overlap. A pinch involves keeping the thumb in a fixed pose and orientation for a certain amount of time. Based on pilot studies, I empirically chose a 800 millisecond window of sensor data to represent the pinch event, provide sufficient signal for feature extraction, and still achieve interactive speeds. For dynamic gestures (i.e., which require movement of the thumb), I use the gyroscope signal for segmentation. The angular velocity measured by the gyroscope when the thumb is stationary is zero. To determine the starting point of the gesture, I use

the combined energy of all gyroscope axes, namely:

$$E(t) = g_x(t)^2 + g_y(t)^2 + g_z(t)^2$$

I empirically set a threshold for E(t) > 400 (from gyroscope °/sec units) based on pilot study data to determine the beginning of an input event. I use a fixed time window (2.2 seconds) which is sufficient to capture all input events for taps, swipes, flicks and gestures. A dynamic window size could be implemented using additional energy signal and length of time thresholds at to determine the end of the gesture.

Classification

The real-time classifier maintains a queue of the segmented windows of data to classify. I utilize the sequential minimum optimization implementation of the SVM in the Weka toolkit[2] trained using a cubic polynomial kernel. Boyer-Moore's linear time voting algorithm [9] is used to determine the gesture performed from a sequence of classifier output labels from frames of sliding window data. The algorithm selects the classifier class label which is a majority and more than half of the sequence. If no such label exists, I select the last label in the sequence. Typically, the classifier settles within seven to nine frames (i.e., less than 2 seconds) of data after a pinch.

4.5 Evaluation I

I conducted two studies in a laboratory setting at our institution to evaluate the recognition of the unique thumb interaction set. The first evaluation focused on recognizing pinches, the second on recognizing taps, swipes, flicks, and gestures.

[2]http://www.cs.waikato.ac.nz/ml/weka/

4.5.1 User Study I: Pinches

To evaluate the system, I recruited 8 participants (4 male, 4 female) from my social network. The ages of the participants ranged from 21 to 34 (mean 26). Seven of the participants were students, and all had some experience interacting with a mobile device using their fingers. Users were compensated $15 for their participation in the study.

Experimental Conditions and Setup

I conducted a user-dependent recognition evaluation, asking each participant to complete three experimental conditions: a four pinch, a five pinch, and a twelve pinch condition in a randomized counterbalanced order (see Figure 4.2). The training phase for each condition was completed before continuing to testing. The input events during all conditions were randomized during training and testing. Participants were asked to sit comfortably with their arm resting on a table in front of them with palm perpendicular to the table, while wearing the Thumby v1 system. A desktop monitor was placed in front of the participants and used to present stimuli. The current setup was designed and piloted for right-handed users and all participants wore the device on the right hand. The thumb pad was placed closest to the base of the thumb and centered relative to the thumb nail.

Procedure

Familiarization: Participants were given a brief description of the research goals and the experimental procedure. The first phase of the study required calibrating the hardware and training participants on the input events. The sensors were calibrated at the beginning of each session of the user study. 2000 samples of each gyroscope axis were averaged to calculate a zero-offset calibration for all channels. The experimenter then helped place the device on the participant's thumb and wrist. Participants were introduced to the task and practiced repeating the demonstration event at least 3 times for each pinch location. This step allowed participants to become familiar with the gesture set prior to beginning

the study. Users were encouraged to maintain their hand in a comfortable position holding their arm steady.

Training: I collected training data in the second phase of the evaluation. The monitor in front of the participants displayed the image of a hand and illuminated the location of the phalange the user should target. A random location was generated by the data collection system. To capture some pose variations from the pinch, each pinch was sampled 10 times taking about 10 seconds. After the sampling was complete, the system randomly prompted the user with the next pinch until each pinch had been sampled twice for the four pinch and five pinch conditions (20 pinch examples) and three times (30 pinch examples) for the twelve pinch condition. Training lasted approximately 2 to 3 minutes for the four pinch and five pinch conditions, and approximately 6 minutes for the twelve pinch condition. At this point, the classifier was trained with the participant's data. The participant was given a chance to practice performing each event correctly at least twice more to familiarize themselves with the testing procedure.

Testing: The final phase of the evaluation was testing the recognizer. I used the training data to build the classifier. The conditions were randomized. Participants were asked to pinch 10 times per location in each condition. During testing, participants were again presented with a visual stimuli of where to pinch. Audio feedback was provided to prompt the user to perform the pinch. Feedback from the classifier was overlay on the hand image in real time. Green color was shown if the classifier output matched the stimulus and red if it did not. Eleven overlapped windows of pinch data were sampled, corresponding to about 5 seconds. The system used a majority voting algorithm to select the predicted event label for that pinch. The system then automatically transitioned to the next stimulus. Finally, a short questionnaire was administered at the end of all three conditions.

4.5.2 User Study II: Taps, Swipes, Flicks & Gestures

The second study was focused on evaluating the recognition performance for taps, swipes, flicks, and gestures. I recruited 6 participants from my social network (4 male, 2 female) ages 19 to 30 (mean 25), 5 of them are students. Users were compensated $15 for their participation in the study.

Experimental Conditions and Setup

I conducted a user-dependent recognition evaluation with two experimental conditions: a single experiment to recognize taps, swipes, and flick events, and a second experiment to evaluate recognition of the “gestures” set. Conditions were randomized. The experimental setup followed the guidelines for User Study I.

Procedure

A familiarization phase, similar to the first study, took place at the beginning of this study.

Training: The set of dynamic input events (e.g., taps, swipes, flicks, etc.) required each input event to be performed and individually segmented by the system, as opposed to sampling windows in the pinches condition. Text on the participants’ screen (e.g., "swipe right", "swipe left", etc.) prompted the user with the action to perform. A total of 10 event examples were collected individually. I randomly collected 5 examples at a time for each input event with a total of 10 examples. Collecting examples for both conditions in small sets (similar to sampling in the pinch condition) helped keep training times low. The training time for both conditions was about 8 minutes. The gyroscope energy feature, the sum of the squares of the sensor axes, was used to automatically segment input events. The experimenter would advance to the next random stimulus once five examples of a particular event had been captured for any given gesture, until all training data was collected. The participant was given a chance to practice for a few minutes with live feedback from the classifier prior to the beginning of the testing phase.

Testing: A fully automated testing phase followed. A stimulus in blue was presented to the user and feedback was shown in green if correctly recognized of in red if not. Segmentation occurred automatically based on the user's motion. The participant was asked to keep their arm and hand as steady as possible to avoid false triggering the system.

4.5.3 Results

In this section, I report classification accuracies and feedback from participants for each of the five experimental conditions: four pinches, five pinches, twelve pinches, taps/swipes/flick, and drawing gestures.

Four Pinches

Eight participants were asked to each pinch 10 times at four locations plus demonstrate an open hand event for a total of 400 gesture examples. Classification accuracy for this condition averaged 90.1% (SD=12.8%, chance=20%) across participants. The confusion matrix is shown in Figure 4.8.

The information transfer rate for Thumby v1 in the four pinches condition (i.e., best condition overall) is 0.69 bps. The ITR is calculated using an interaction time of 2 seconds (i.e., 0.5 interactions per second), accuracy of 90.1%, and 4 total pinch gestures.

I wanted to test the reproducibility of the recognition between sessions, that is, whether users could remove the ring and put it back on without retraining. Four of the participants from the four pinches condition were asked to participate in an additional experiment to measure reproducibility. The ring device was removed between the 3 sessions and the original training data was used. Mean accuracies (97.0%, 97.5%, 94.5%) across participants did not exhibit any significant decrease. The results suggest that calibration may be unnecessary as the ring's physical design provides guidance on proper sensor placement on the thumb. Out of the four phalanges and open hand, the ring distal exhibited the lowest overall accuracy at 78.1%.

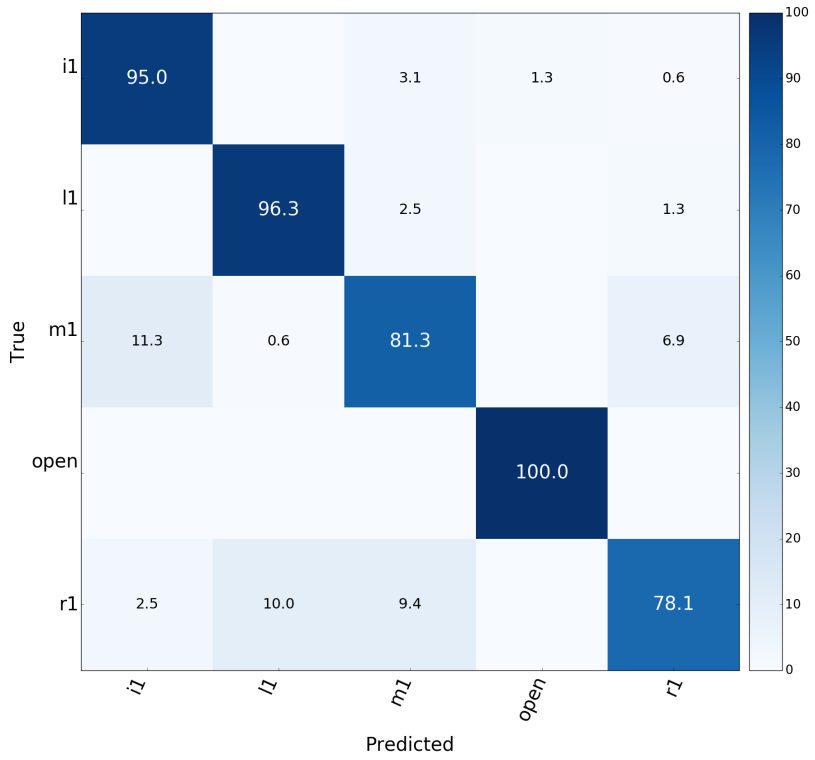


Figure 4.8: Confusion matrix for the four pinches condition.

Five Pinches

Eight participants were asked to each pinch 10 times at five locations and demonstrate an open hand event for a total of 480 gesture examples. Classification accuracy for this condition averaged 89.4% (SD=7.4%, chance=16.7%) across participants. Out of the five phalanges, the ring medial exhibited the lowest overall accuracy with 81.3% and the middle distal had the highest overall accuracy with 97.5%. The confusion matrix is shown in Figure 4.9.

Twelve Pinches

Eight participants each pinched at twelve pinch locations and performed an open hand pose 10 times for a total of 1040 unique events. The accuracy was tested using three rounds of training data. Classification accuracy for twelve pinches averaged 70.3% (SD=17.0%, chance=7.7%). The little distal exhibited the lowest overall accuracy with 47.5% confused

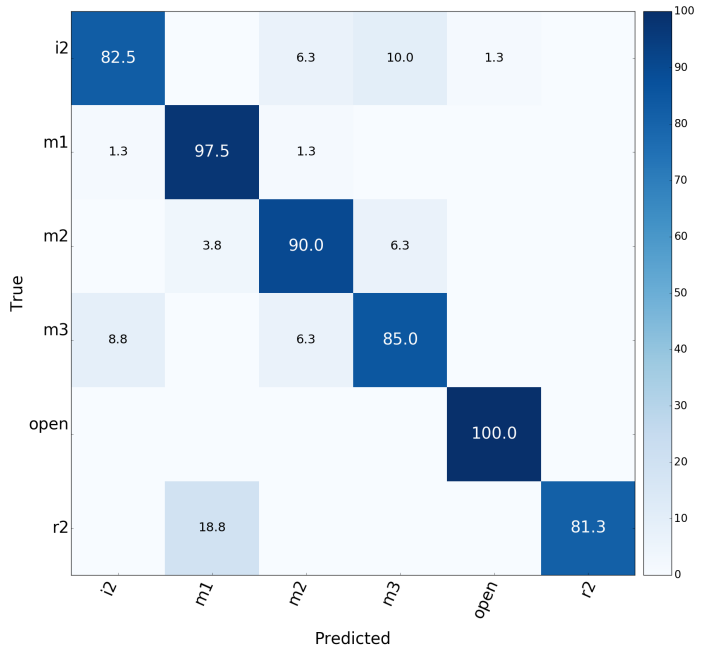


Figure 4.9: Confusion matrix for the five pinches condition.

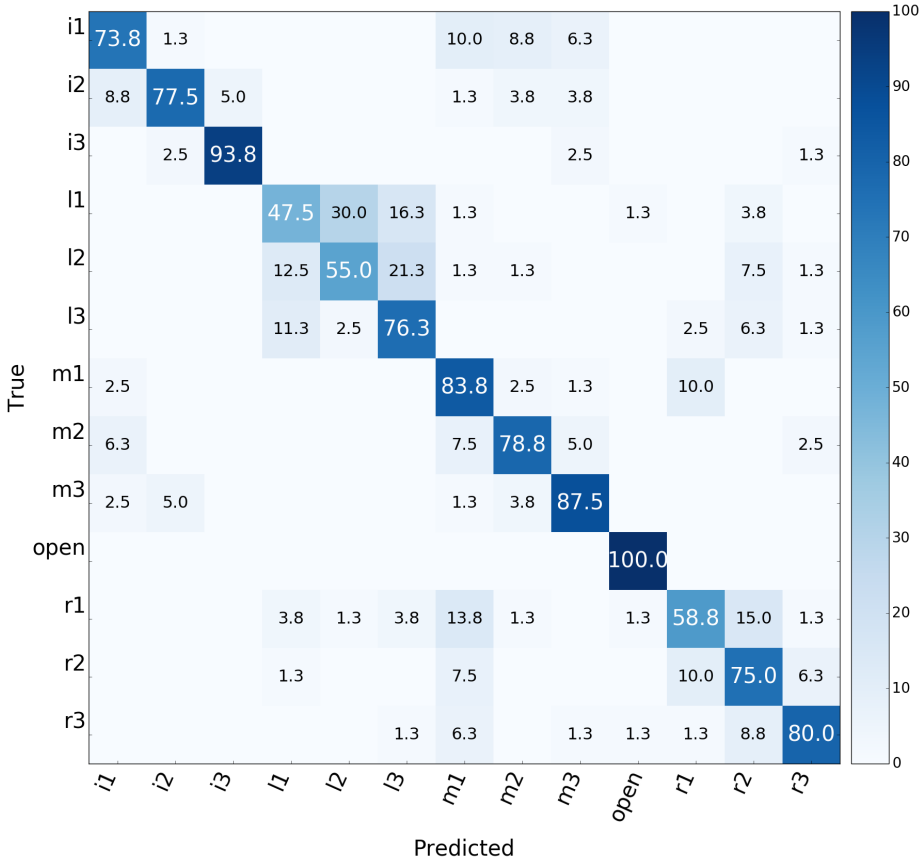


Figure 4.10: Confusion matrix for the twelve pinches condition.

with its adjacent little medial. The index proximal location exhibited the highest accuracy and was the target of the right tap and double right tap event. The confusion matrix is shown in Figure 4.10.

Taps, Swipes, Flick

Taps, swipes, and flick events were trained and classified with a single recognizer. Six participants each performed nine distinct event 10 times for a total of 540 events. Classification accuracy for this condition averaged 84.6% (SD=6.8%, chance=11.1%) across participants. The confusion matrix is shown in Figure 4.11.

Gestures

All gestures drawn on the surface of the phalanges were trained with a single recognizer. Six participants each performed eight distinct events 10 times for a total of 480 events. Classification accuracy for this condition averaged 70.2% (SD=5.8%, chance=12.5%) across participants. The confusion matrix is shown in Figure 4.12.

Questionnaire and Qualitative Feedback

Participants were asked to complete a short exit survey about their experience with the Thumby v1 system during the study and potential use of the device outside the laboratory. The factors I evaluated for the Thumby v1 system included comfort (1 very uncomfortable, 7 very comfortable), ease of use (1 very hard, 7 very easy), and ease of learning (1 difficult to learn, 7 easy to learn). The results are shown in Table 4.2.

Comfort	Ease of Use	Ease of Learning
4.4 (SD=1.5)	5.0 (SD=1.5)	6.3 (SD=1.0)

Table 4.2: Average results of questionnaire with 14 participants using a 7-point Likert scale, rating comfort, ease of use, and ease of learning.

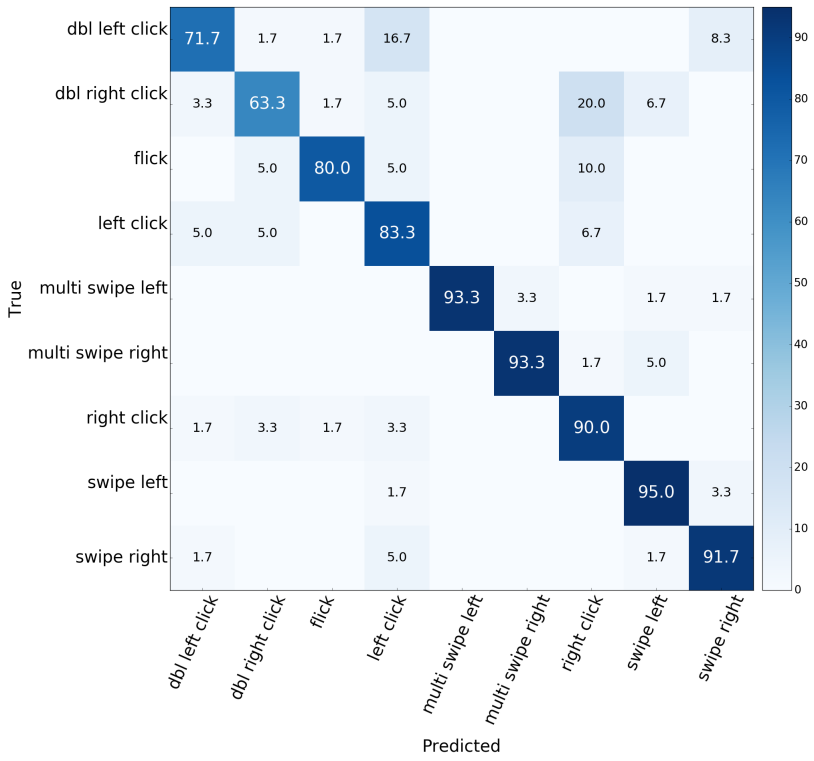


Figure 4.11: Confusion matrix for the taps, swipes, and flick condition.

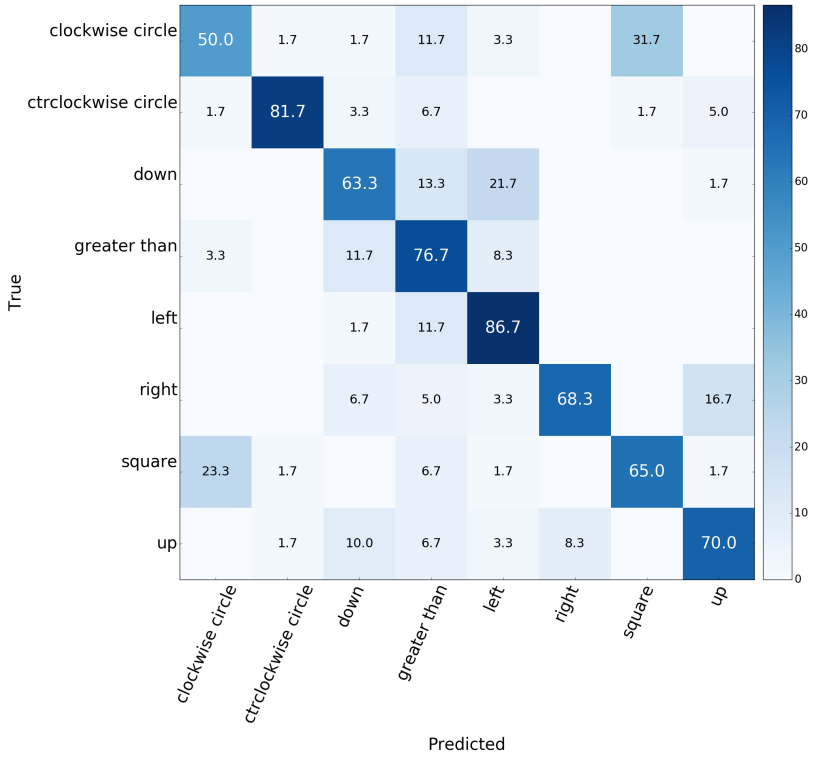


Figure 4.12: Confusion matrix for the palm drawing gestures condition.

4.6 Discussion

Pinching the phalanges represents a unique challenge from a sensing perspective as the phalanges (i.e., distal, medial, proximal) are located in near proximity. The system is able to recognize the four pinches in real time with 84.3% accuracy. As a comparison point, WristFlex uses pressure sensors to detect pinching the fingertips with 80.5% [22] during live testing. Skinput on the other hand uses bio-acoustic sensing and reported 87.7% accuracy for tapping the fingertips [42]. While those results are comparable to Thumby v1, our system enables thumb pinches with a much simpler system (using only a single accelerometer and gyroscope).

A natural extension to the four and five pinch scenarios is to increase the number of possible target locations. This would enhance the value of the Thumby v1 system with a new range of potential applications that require a larger number of discrete events. The twelve pinch confusion matrix (Figure 4.10) shows that most classification errors occur with the little finger. However, from the four pinch and twelve pinch condition, I observe confusion between the ring distal and adjacent little and middle distal phalanges. The confusions may be due to sensors shifting or participants' under/over-extension of the thumb, which the system currently does not account for. When asked in the exit questionnaire, participants reported difficulty reaching the tip of the little finger with the thumb, and researchers observed participants rotating the wrist in some cases to compensate for this challenge.

The open hand was detected with nearly 100% accuracy across all conditions suggesting that perhaps the event could be used as a rest state upon entering gesture mode. For the taps, swipes, and flicks, I observed that events involving repetitive movement such as multi-swipes were most robust to classification errors. However, interestingly, the most confused interaction was observed with double right tap and right tap. I observed this confusion occur mostly in situations where soft taps occurred and the system was unable to distinguish the event. Finally, for gestures, I observed the most confusion with the clockwise and

square gestures due to the similarity in the direction and shape of the gesture. Designing a gesture set that is more easily distinguishable would likely help improve results.

4.7 Example Interfaces and Interactions

To illustrate the utility of using Thumby v1 as a wearable input device with disaggregated devices (e.g., head-mounted displays, mobile phones, smart watches, and smart home devices), I demonstrate a variety of applications that take advantage of the gesture set. I describe several example applications below.

4.7.1 Smartwatch Music Player

To demonstrate the pinch interaction technique using the fingertips, I created a basic music player on a smart watch. The button controls are aligned horizontally along the fingertips: a pinky pinch for previous song, a ring finger pinch for play/pause, a middle finger pinch for stop, and index finger pinch for next song. Swipe gestures are used within the application to control the volume or fast forward within the song.

4.7.2 D-Pad Controller

Directional pads are four-way directional controls (up/down/left/right) typically found on game controllers and remote controls. I used five different phalanges to enable the functionality of a D-Pad using Thumby v1.

Thumby v1 can serve as a complete replacement for interaction on an Android Wear smart watch. I used the five phalanges pinch layout to control all the currently available interactions on Android Wear, namely swipe up, swipe down, swipe left, swipe right, and tap. Thumby v1 allows users to take full control of their smart watch without having to touch the device's smaller screen, which is useful in the case of hands busy situations. The interactions enabled by Thumby v1 could enable eyes-free operation of the device for quick actions as well, such as muting a phone call.

The D-Pad can also provide a useful input modality for menu navigation on a head-mounted display. I demonstrate the use of Thumby v1 to interact with a head-up display which typically require bringing the hand to the side of the head for input. Using Thumby v1, a user is able to navigate the through cards on the timeline using left and right commands, select cards and actions. The up command can be used to dismiss an action or return to the previous card. The down command could be as a shortcut to take a picture or make a phone call.

4.7.3 Alphanumerical Input

A 12-button keypad has been used for text-input on various devices. Each key is letter-mapped with a maximum of 4 letters. The full functionality can be extended with the two extra keys used for adding space, return and special characters. I used the set of 12 thumb-to-phalange pinches to demonstrate a very basic text-entry system using T9.

4.7.4 Quick Access Shortcuts

Thumby v1 can also be used in conjunction with a smartphone, in particular for situations when the phone isn't immediately accessible in a pocket or bag. I also demonstrate quick access to shortcuts to answer a call or adjust volume settings. I envision there could be a number of scenarios in which an eyes-free, rapid input device could be useful. For example, lifelogging or quantified self to record a timestamp specific event (e.g., exercising, eating).

4.8 Limitations of Standalone Thumb Ring

One of the challenges of building mobile and wearable systems is enabling these devices to operate across users in a variety of situations. Upon completion of the evaluation I prototyped the addition of a 6 DoF IMU at the wrist to explore: 1) capturing arm orientation relative to ground during input, and 2) to use arm movement to activate the ring-based interface. Typically, a button on the ring would be used to initiate the system, similar

to the Ring commercially available product. Understanding the arm pose in unison with thumb-ring events may easily increase the input vocabulary recognized by the system.

Thumby v1 relies entirely on inertial sensors leading to a practical concern about everyday use. An activation gesture to trigger the input mode may address false positives. I have begun exploring a double flip of the wrist motion to activate the system, inspired by DoubleFlip [95]. Evaluating the activation gesture and performance of the recognizer in different arm poses remains as future work. I envision in the future that the thumb-ring could be paired with a smartwatch on the same hand to achieve these capabilities. I explore this idea with Thumby v2.

Earlier I described the vision and design space for a wearable, one-handed, eyes-free input device. Thumby v1 is the initial embodiment of this vision using inertial sensing in a thumb-ring form factor. The system is capable of detecting over 20 unique interactions for microinteractions on smartwatches, head-mounted displays, and other connected devices. However, the system suffers from several limitations. First, the energy-based segmentation is sensitive to slight movements and is not robust to everyday use (i.e., while walking). Possible improvements would be the introduction of an activation gesture or a context-based segmentation based on device and application use. Additionally, the single point of contact sensing on the thumb does not enable measurements relative to a reference point and depends on avoiding gravity vector shifts.

In the next iteration of the system dubbed Thumby v2, I shift the interaction focus from head-up displays and connected devices to smartwatches. I also address the challenges related to segmentation, recognizer performance, and robustness for everyday use.

4.9 The Thumby System - Version 2: Smartwatch + Thumb Ring

The second iteration of Thumby uses 9 DoF inertial sensors embedded in a thumb-worn ring and 9 DoF sensing in a wrist-worn smartwatch to provide one-handed and quickly accessible interaction capabilities to the smartwatchs. This version of the system seeks to

address the segmentation, performance, and robustness challenges by: 1) segmenting using an automatically triggered 2-second window once the wrist is raised; 2) upgrading the 6 DoF IMU thumb ring to a 9 DoF IMU with absolute device orientation; and 3) introducing the smartwatch at the wrist with its 9 DoF IMU and absolute orientation.

In this section, I describe the second iteration of the system named Thumby v2, interaction techniques it supports, the hardware prototype, data pipeline, and technical evaluation.

4.9.1 Interaction Techniques

Introducing the smartwatch provides an immediate glanceable display and sensing at the wrist that can be used along with the Thumby v2 thumb ring. Motivated by the need for input to the smartwatch, the system provides a variety of natural interactions while the user wears the device on their left hand. I take advantage of the dexterity of the thumb and the wrist. The system provides the ability to detect 8 gestures - index finger single and double taps, a ring finger tap, thumb swipes left and right, thumbs up and down, and a rest pose where the hand and fingers are kept open and still. These gestures were chosen as a subset of the gestures used in Thumby v1 and informed by the taxonomy described in Chapter 3. The natural pose of the arm during all interactions is raised at chest level with the device in direct line of sight.

Taps

Taps and double taps provide an additional vocabulary of input symbols, and are useful for interactions such as binary state dialog boxes, selection, and shortcuts. The system recognizes taps of the thumb against the tip of the index and ring finger, and a double tap of the index finger.

Swipes

Swipes are useful for indicating increasing or decreasing changes of value, such as flipping through arrangements of items (e.g., list items, pictures) or directional state changes (e.g., fast forward, previous song, or increasing/decreasing volume). The top lateral part of the extended index finger is used as a surface area for thumb swipes. I explore 2 distinct swiping gestures: swipe left-to-right and right-to-left.

Thumbs Up and Down

A thumb gesture, usually described as a thumbs-up or thumbs-down, is a common hand gesture achieved by a closed fist held with the thumb extended upward or downward in approval or disapproval, respectively. In this case, the gesture is characterized by the motion of the wrist (i.e., smartwatch) and the thumbs (i.e., thumb ring) in either direction. These gestures are commonly used as metaphors in English, as well as prevalent in social media and online recommendation systems. On a smartwatch, a thumbs-up or thumbs-down could be used to quickly acknowledge or dismiss a notification, or perhaps express a positive or negative feeling in response to a question over SMS.

Rest

A rest pose is characterized by the user rotating the wrist and smartwatch toward the user and keeping the fingers and thumb extended. The interface is designed and classifier is trained to treat the rest gesture as a non-event or a null class gesture. For example, the user receives a notification and raises their arm to line of sight but does not perform any of the gestures described above.

4.9.2 Hardware Prototype

The Thumby v2 prototype is a thumb-worn ring with a MbientLab MetaMotionR 9-axis IMU (accelerometer, gyroscope, magnetometer) and an on-board sensor fusion module

with 8MB Flash Memory (see Figure 4.13). The system is designed to capture movements of the thumb and contact with the fingers. The accelerometers measure the relative force due to gravity based on the orientation of the sensor relative to ground, while the gyroscopes measure angular rotation. The sensor fusion uses the accelerometer, gyroscope, and magnetometer to output the absolute orientation of the device in world coordinates, expressed as a quaternion.

The thumb-worn IMU is battery-powered and wirelessly connected to a Sony Smartwatch 3 over Bluetooth running Android Marshmallow. The Android SDK provides access to the smartwatch accelerometer, gyroscope, and absolute device orientation as a quaternion. The thumb-worn IMU is mounted inside of a 3D-printed case with a concave curvature to sit comfortably on the thumb. The curvature also ensures that the device is centered on the top of the thumb, which is important for device usage over time (e.g., if the ring is removed daily or tends to shift during daily activities). For a full depiction of the hardware, see 4.13.

The hardware specifications of the system include:

- 1 x 9 DoF IMU board MetaMotionR from MbientLab. The IMU board provides a 6-axis BMI160 accelerometer+gyroscope with a range of $\pm$16g and full-scale range of $\pm$2000°/sec, respectively. The module also uses the on-board 3-axis BMM150 magnetometer and Bosch sensor fusion module for absolute device orientation. The board is built around the nRF52 SOC from Nordic using a ARM Cortex M4F CPU and Bluetooth Low Energy.

- 1 x 100mAH lithium-ion 3.7V battery connected to the thumb ring.

- 1 x Sony SmartWatch 3 SWR50 running Android 6.0.1 with a 3-axis accelerometer, 3-axis gyroscope, and absolute device orientation using sensor fusion and the 9 DoF IMU with magnetometer.

Figure 4.13: The Thumby v2 system includes a 9 DoF IMU (left) worn around the thumb in a 3D-printed case connected to an Android Sony SmartWatch 3 SWR50 (right).

4.9.3 Recognition Pipeline

In this section, I describe how the raw sensor data is turned into input events: sensor data acquisition, preprocessing and filtering, sensor synchronization, feature extraction, and gesture classification.

Segmentation

This phase focuses on determining when an event of interest occurs within the sensor stream. For the offline recording purposes of the training, users are prompted to provide input and sensor data is recorded for each gesture individually in 2-second windows. In practice, the segmentation window could similarly be triggered based on a notification-response or other push-to-gesture activation technique.

Sensor Data Acquisition

The 9 DoF thumb ring provides accelerometer, gyroscope, and absolute device orientation (quaternion) using an on-board sensor fusion module and is delivered to the watch via Bluetooth. The effective sampling rates are approximately 40 Hz, 40 Hz, and 20 Hz, respectively. The Sony SmartWatch 3 SWR50 provides accelerometer, gyroscope, and ab-

solute device orientation (quaternion) using the Android SDK. The Android SDK's highest sampling rate (SENSOR DELAY FASTEST) is used for the watch with effective sampling rates of approximately 200 Hz, 200 Hz, and 100 Hz for accelerometer, gyroscope, and absolute orientation, respectively.

Sensor data is stored in memory during the 2-second acquisition window and asynchronously written to file once the gesture is complete. Each device and sensor data source is written to separate files. The watch sensor data recorded includes the system timestamp (Unix time), the sensor timestamp (nanoseconds since application uptime), the sensor values for each axis, and a line counter for each sensor event. The ring sensor data recorded includes the system timestamp (Unix time) when the data arrives in the application, the Bluetooth timestamp when the sensor data arrives on the Bluetooth stack, the sensor values per axis, and a packet counter for each event delivered.

Sensor Data Synchronization

To facilitate accurate gesture recognition, the two data streams from the thumb ring and smartwatch must be synchronized.

The two data sources are synchronized using timestamps provided by the ring and watch sensors. The watch timestamps are based on the sensor time (in nanoseconds) with the initial system time (in milliseconds) as a starting point. The ring timestamps are based on the time of arrival (in milliseconds) on the Bluetooth stack.

First, both the ring and watch sensor data are trimmed to remove any non-overlapping early or late sensor events and to find a time interval where both data sources overlap. Following this alignment process, the two data sources are combined by resampling the ring and watch data and combining the results. A downsampling rate of 20 Hz is chosen based on the minimum sampling rate of the thumb ring and to accommodate uneven sampling rate between the ring and watch data sources. Finally, the data is resampled using interpolation and is combined to produce a data structure with equal number of samples for all sensors.

Relative Device Orientation Using Quaternions

With knowledge of the complex number system and complex planes, it is possible to extend this knowledge to 3-dimensional space by adding two imaginary numbers to the number system in addition to i.

Quaternions are an extension of the complex numbers to four-space, and are represented as algebraic quantities with three orthonormal imaginary axes (i, j, k).

The general form to express quaternions is:

$$q = w + xi + yj + zk$$

where w, x, y, z $\epsilon\mathbb{R}$.

According to Hamilton's famous expression:

$$i^2 = j^2 = k^2 = ijk = -1$$

$$ij = k \quad \text{and} \quad jk = i \quad \text{and} \quad ki = j$$

$$ji = -k \quad \text{and} \quad kj = -i \quad \text{and} \quad ik = -j$$

For the purposes of this work, a quaternion is simply a four-vector that represents the absolute orientation of a device in free-space. It inherits all vector properties and operations, including dot product, scalar multiplication, addition and norm.

A quaternion is represented as a set of 4 real numbers [x y z w] which represent rotations the following way:

$$x = RotationAxis.x * sin(RotationAngle/2)$$

$$y = RotationAxis.y * sin(RotationAngle/2)$$

$$z = RotationAxis.z * sin(RotationAngle/2)$$

$$w = cos(RotationAngle/2)$$

The rotation angle is expressed in radians. The RotationAxis is, as its name implies, the axis around which the device rotates. RotationAngle is the angle of rotation around this axis. Essentially, quaternions store a rotation axis and a rotation angle, in a way that makes combining rotations easy.

One of the most important reasons for using quaternions in computer graphics is that quaternions are very good at representing rotations in space. Quaternions overcome the issues that plague other methods of rotating points in 3D space such as Gimbal lock which is an issue when you represent your rotation with Euler angles.

The quaternion representation allows for better handling of gimbal lock and smooth interpolation of angles. For a full description of quaternions and more details, see the Quaternions and Motion Interpolation: A Tutorial textbook by Heise et al. [44].

The 9 DoF thumb ring hardware and the smartwatch enable the measurement of absolute device orientations (quaternions) for both devices in world coordinates. Using the two orientations, it is now possible to calculate the relative orientation changes between the ring and the watch during a gesture, using the watch as a reference.

Using quaternions, it is possible to define several methods that represent a rotational interpolation in 3D space. SLERP stands for Spherical Linear Interpolation. SLERP provides a method to smoothly interpolate a point about two orientations. I use the transformations.py library[3] to synchronize and align the quaternions of the thumb ring and the watch.

Once the ring and watch quaternions are synchronized, the rotation of the ring relative to the watch is calculated. The "subtraction" of the watch quaternion from the ring quaternion is equivalent to computing the angular difference between the two quaternions. This is accomplished by quaternion multiplying the ring quaternion with the inverse of the watch quaternion.

[3]http://www.lfd.uci.edu/ gohlke/code/transformations.py.html

The final representation of the orientation is expressed as a unit vector. A rotation matrix is generated from the quaternion then used to transform a 3-dimensional unit vector. The x, y, and z components of the transformed vector are used as features.

Feature Extraction

I characterize the gesture set using 450 features and offline cross validation. I rely on the empirical cumulative distribution function (ECDF) of the accelerometer and gyroscope for each of the smartwatch and thumb ring, as well as the ECDF representation of the delta quaternion between the ring and the watch.

The ECDF representation provides an objective measure to effectively capture and preserve the distribution of the sensor data for each window, typically lost in certain statistical features (e.g., mean). In particular, the spatial position and general shape of the representation are useful in distinguishing events from each other. I refer to Hammerla's work for a more detailed description and implementation of the ECDF [36].

Classification

I use a support vector machine (SVM) algorithm trained using Weka's sequential minimal optimization (SMO) implementation with a polykernel and default parameters.

4.10 Evaluation II

4.10.1 Study Design

I conducted a technical evaluation of the interaction techniques with an unmodified Sony SmartWatch 3 in the usability lab of the institution. Eight participants (5 male, 3 female, ages 19-60, students) were part of the user study and were compensated $10 for their participation. One participant identified as ambidextrous and all others were right handed.

The within-subjects study was designed for two conditions (sitting and walking) and a total of 8 gestures (see section above). A 2x2 Latin square design was used to counterbal-

ance the sitting and walking conditions. Participants were randomly assigned to a row of the Latin square.

Participants wore the watch on the left hand. To begin, researchers performed a demonstration of each gesture. Participants familiarized themselves with the data collection application and practiced the gestures with at least one practice round, or additional rounds as needed based on the assessment of the researcher conducting the study. The study was designed to collect sensor data for offline analysis and no live feedback was provided.

During the study, the participants were asked to keep their arm and hands on their lap (while sitting) and down by the side of their hip (while walking). A haptic vibration was delivered randomly to the user prompting them to raise their arm to chest level and line of sight. Once the arm was raised after a 2 second delay, a visual text display was presented to the participant on the watch screen prompting them which gesture to perform. A progress bar was used to indicate the time elapsed for recording. Thumb ring and watch sensor data was recorded over 2 seconds and stored on the smartwatch across 6 different files for each device's accelerometer, gyroscope, and absolute orientation. Each gesture was recorded in individually labeled files for segmentation. Once the participant performed the gesture and the progress bar expires, they are asked to lower their arm to a resting position and await the next prompt. Participants are allowed to rest, drink water, remove the watch, or leave the room between rounds if desired.

For the sitting condition, the person sat in an office chair in the usability lab of our institution. A swivel chair was used allowing the user to spin around in their chair corresponding to different absolute orientations in world coordinates. Participants performed four rounds of data collection. In each round, participants performed 5 examples for 8 gestures. The order of the gestures was randomized across each round. In summary, the sitting condition of the user study includes 8 participants x 8 gestures x 4 rounds x 5 samples per gesture for a total of 1280 gesture samples.

For the walking condition, the participant walks at normal pace on the second floor of

our institution's research center. A research walks side by side and engages in conversation. No special walking track is used and obstacles such as other people or objects in the path are not controlled for. The goal is to record natural walking. Participants perform four rounds of data collection. In each round, participants perform 5 examples for 8 gestures. The order of the gestures is randomized across each round. In summary, the walking condition of the user study includes 8 participants x 8 gestures x 4 rounds x 5 samples per gesture for a total of 1280 gesture samples.

I evaluate how the system performs on a per-user level using 10-fold cross validation and how it generalizes across participants using leave-one-participant-out analysis.

4.10.2 Per-User Classifiers

I evaluate the gestures by applying 10-fold cross validation on each individuals' collected instances with a SVM polykernel. For the sitting condition, the overall average per-user accuracy across 8 participants and 8 events is 90.7% (sd=8.4%). P1 achieves the highest accuracy at 96.3% and P4 achieves the lowest accuracy at 81.9%. The confusion matrix of the results is shown in Figure 4.14. The lowest precision of 85.0% is observed for the double tap index, mostly confused with a single tap index. In some cases, participants perform the double tap lightly, effectively being recognized as a single tap index. The swipe right event achieves the highest accuracy at 96.3%.

The information transfer rate for Thumby v2 in the sitting condition (i.e., best condition) is 1.15 bps. The ITR is calculated using an interaction time of 2 seconds (i.e., 0.5 interactions per second), accuracy of 90.7%, and 8 total gestures.

For the walking condition, the average accuracy using 10-fold cross validation across 8 users and 8 events is 84.6% (sd=8.9%). P2 achieves the highest accuracy at 88.8% and P5/P8 both achieve the lowest accuracy at 78.1%. The confusion matrix of the results is shown in Figure 4.15. The lowest precision of 78.1% is observed for thumbs up. The swipe left gesture is the most accurate with an accuracy of 89.4%.

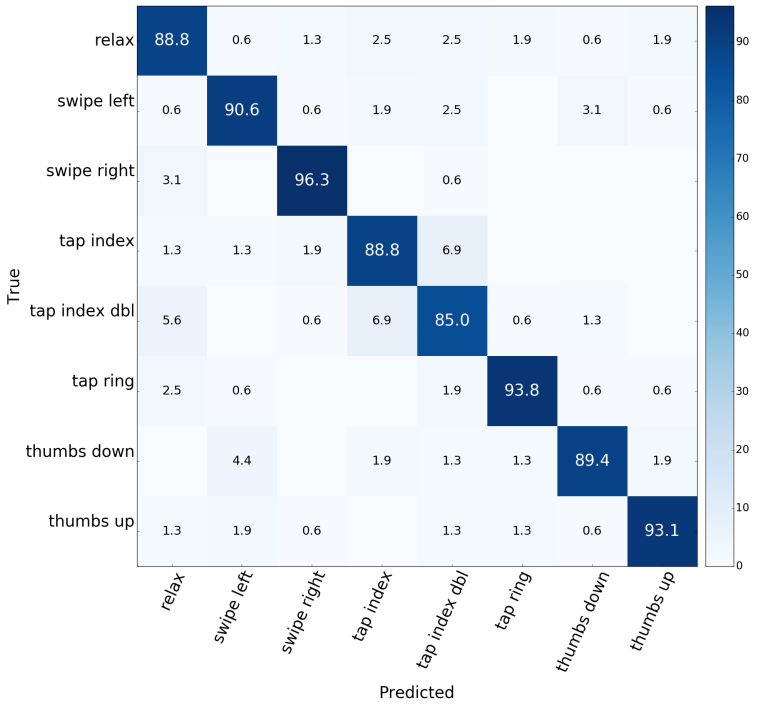


Figure 4.14: User dependent confusion matrix averaged across all users (in %) for the sitting condition. Rows represent ground truth and columns are predicted values.

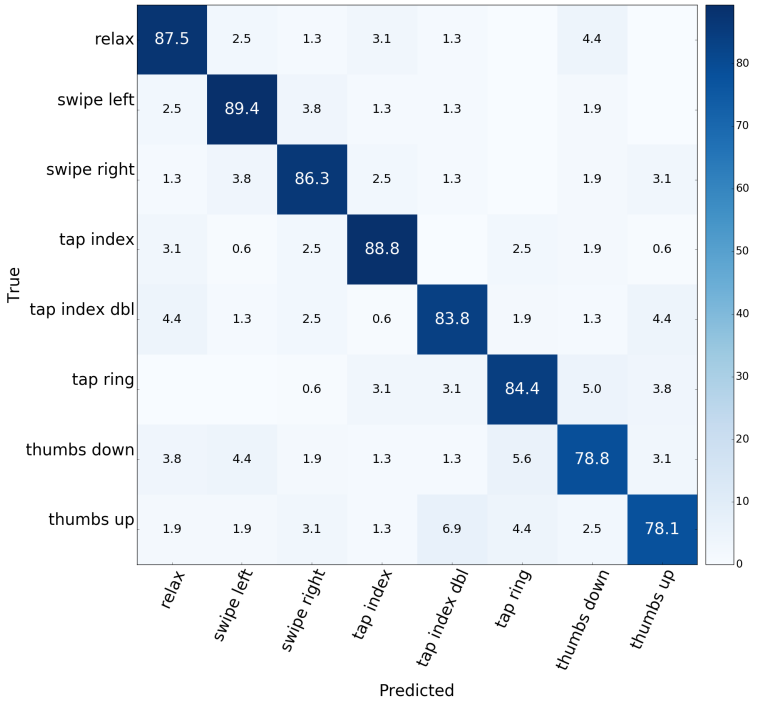


Figure 4.15: User dependent confusion matrix averaged across all users (in %) for the walking condition. Rows represent ground truth and columns are predicted values.

4.10.3 Best Case Per-User Classifiers

I re-evaluate the classification results with a subset of all gestures to find the best case models for sitting and walking. I calculate all combinations of the gesture labels (i.e., 8 choose 3) to find a subset of three optimal events and perform 10-fold cross validation with the resulting labels only. In general, there were multiple combinations with comparable accuracy so I report a combination of gestures that seem to make most sense when used together. The best accuracy for sitting is 99.4% (sd=4.1%) achieved for the swipe left, swipe right, and thumbs up gestures. The combination of these gestures could be used to navigate a 2-dimensional interface with swipes and selection with a thumbs up. The best accuracy for walking is 99.2% achieved for the relax, swipe left, and thumbs up gestures.

4.11 Smartwatch Example Interfaces and Interactions

To illustrate the utility of using the second iteration of Thumby as a wearable input device with a smartwatch, I present a set of envisioned applications that take advantage of the gesture set.

4.11.1 Music Player

A basic music player on a smartwatch could take advantage of pinch interactions. The button controls could be aligned horizontally along the fingertips: a pinky pinch for previous song, a ring finger pinch for play/pause, a middle finger pinch for stop, and an index finger pinch for next song. Swipe gestures could be used to control the volume or fast forward within the song.

4.11.2 Notification Response

A notification response interface describes an interface in which a gesture is used to respond to an incoming notification (e.g., phone call, text message, etc.). For example, a quick

thumbs up or thumbs down could be used as quick confirmation to a question received via text message or to reply with an emoji.

4.11.3 Quick Command

A thumbs up or down could also be used to confirm an action performed using a different modality. For example, canceling a text message transcribed from a voice command.

4.11.4 Quick Access Shortcuts

An index finger double tap event can be used as an activation gesture for the smartwatch to launch a quick access shortcut. The shortcut could be used to start a timer while working out or start the music player.

Given the smartphone is already connected via Bluetooth to the mobile phone, an index finger tap event could be useful for situations when the phone isn't immediately accessible in a pocket, bag, or on a tripod. The user could remotely trigger snapping a picture with their smartphone camera without touching the smartphone.

4.12 Contributions

I present a summary of the Thumby system and its contributions using this dissertation's research goals. Figure 4.16 presents a summarized version.

Goal 1: Achieving *expressiveness* through a spectrum of interaction events and wearable interfaces

- Type of Interface and Purpose of Interaction

  Thumby is a finger-level motion gesture interface. Inertial sensing of the wrist and thumb eliminates smartwatch occlusion and provides a broad set of notification-response and command gestures.

- Number of Unique Input Actions

  A total of 37 gestures are supported. The first iteration of Thumby supports up to 12 thumb-to-phalange pinches, 8 drawing thumb gestures on the palm of the hand, and 9 taps/swipes/flick gestures. The second iteration supports 8 gestures including taps, swipes, and other thumb-and-wrist gestures.

- Information Transfer Rate

  The information transfer rates range between 0.69 to 1.15 bps for Thumby v1 and Thumby v2, respectively.

- Applications

  I've built a set of applications for Thumby v1: controlling music on a smartwatch, a navigational controller for Google Glass, and alphanumerical input app using T9. Other envisioned applications are detailed for smartwatch input with Thumby v2.

Goal 2: Enabling rapid *speed* of interaction and reducing the time between intention and action

- Time Between Intention and Action

  The time between intention and action (i.e., setup time) is <0.5 seconds for Thumby v1 as the user can perform gestures without looking at the wrist or thumb. For Thumby v2 using the smartwatch, the user may wish to perform the gesture without looking at the device within 0.5 seconds or raise the arm to field of view resulting in up to 2 seconds of setup time.

- Speed of Interaction

  The interaction can be completed within approx. 2 seconds. Recognition is performed using a segmented window of sensor data and analysis off-device, both offline and online.

Goal 3: Expanding the *practicality* of one-handed input techniques and form factors

- Level of Instrumentation

  Level of instrumentation is high. An active inertial sensor is worn on the thumb and paired with either a laptop (Thumby v1) and a smartwatch (Thumby v2) to capture the absolute orientation and movement of the thumb and wrist.

- Prototype Readiness for Deployment

  Two iterations of a thumb ring device: 1) wired thumb ring requires an external laptop for processing (TRL 4 for a small scale prototype used in the laboratory), and 2) a battery-powered and wireless inertial thumb ring pairs with a smartwatch in a slightly large form factor (TRL 5 for a large scale prototype used in the intended environment).

Goal 4: Building *robust* detection approaches for multiple users and everyday activities

- Accuracy of Technique

  Thumby v1 achieves 90.1% for 4 pinches. Thumby v2 achieves 90.7% for 8 thumb gestures. Thumb-to-fingertip pinches and thumb gestures with thumb- and wrist-worn sensors achieve the highest interactive accuracy rates. Other interactions and accuracies are described in the chapter.

- Usage Across Multiple Users

  The system is currently user-dependent. Classification is robust with individual per-user training. User-independent models are not stable across users (< 30% accuracy) with the current data set and features used.

- Support During Everyday Activities

  Low to Medium support. Thumby v1 (low) results are reported using an online version of the system and is evaluated while sitting only. Thumby v2 (medium) results are based on offline analysis and further work is required to build an online system. Thumby v2 classification is robust to various levels of global motion during sitting and walking, using a combination of wrist- and thumb-worn inertial sensors.

4.13 Summary

In this chapter, I have presented an exploration of inertial sensing to enable a set of finger gestures as an alternative input method for mobile and wearable computing systems. I describe a wearable inertial sensor at the thumb to recognize and identify thumb pinches, taps, swipes, flicks, and gestures. Thumby v1 uses ubiquitous inertial sensors embedded in a thumb-worn ring to provide one-handed, quickly accessible, gloveless, and eyes-free interaction capabilities to connected devices. The approach relies on humans' proprioceptive abilities, which provide awareness of one's body position. Results from experiments with participants demonstrate the accuracy of the system for detecting 4 and 5 thumb pinch locations with overall accuracy of 90.1% and 89.4% respectively, as well as the 12 pinch locations with overall 70.3% accuracy. The system was also able to recognize taps, swipes and flicks with 84.6% accuracy, and gestures drawn on the hand with 70.2% accuracy. A second hardware iteration of the system, Thumby v2, enables a total of 8 gestures — various thumb-to-index finger taps, thumb-to-ring finger taps, thumb swipes, thumbs up and down, and a relaxed hand pose. The system achieves over 90% accuracy while sitting and 85% while walking. Finally, I demonstrate and envision a variety of meaningful input scenarios for mobile and wearable users wearing the sensor-enabled thumb ring. Thumby addresses research goals focused on *expressiveness, speed, practicality* and *robustness*, and its contributions are summarized in Figure 4.16.

Dimensions	Thumby v1	Thumby v2
Goal 1: Achieving expressiveness through a spectrum of interaction events and wearable interfaces		
Purpose of Interaction	Command Notification Response	Notification Response
Number of unique input actions	4,5,12 thumb pinches 8 palm drawing 9 taps, flicks, swipes	8 finger-level thumb gestures
Information transfer rate	0.69 bps	1.15 bps
Applications		Envisioned
Goal 2: Enabling rapid speed of interaction and reducing the time between intention and action		
Time between intention and action	< 0.5 seconds	< 0.5 to 2 seconds
Speed of interaction	Approx. 2 seconds	Approx. 2 seconds
Goal 3: Expanding the practicality of one-handed input techniques and form factors		
Level of instrumentation	High	High
Prototype readiness	TRL 4 Small scale prototype (laboratory)	TRL 5 Large scale prototype (intended environment)
Goal 4: Building robust detection approaches for multiple users and everyday activities		
Accuracy of technique	90.1% for 4 pinches (sitting, online)	90.7% for 8 gestures (sitting, offline)
Usage across multiple users	User-dependent	User-dependent
Support during everyday activities	Low	Medium

Figure 4.16: Summary of contributions and research goals for Thumby v1 and v2.

CHAPTER 5

SYNCHROWATCH: ONE-HANDED SYNCHRONOUS SMARTWATCH GESTURES USING AUTO-CALIBRATION AND MAGNETIC SENSING

5.1 Abstract

SynchroWatch is a one-handed interaction technique for smartwatches that uses rhythmic correlation between users' thumb movement and on-screen blinking controls. The technique uses magnetic sensing to track the synchronous extension and reposition of the thumb, augmented with a passive magnetic ring (see Figure 5.1). The system measures the relative changes in the magnetic field induced by the given thumb movement and uses a time-shifted correlation approach with a reference waveform for sync detection. I evaluated the technique during three distraction tasks with varying degrees of hand and finger movement: active walking, browsing on the computer, and relaxing while watching online videos. Offline analysis results suggest separable correlation coefficients between synchronous gestures versus null data. A second evaluation using a live implementation of the system pipeline and interface running on a smartwatch suggests the usage of this technique is possible for notification response and command gestures. Finally, I developed three demonstration applications that highlight the technique running in real-time on the smartwatch.

5.2 Introduction

The emergence of smartwatches has enabled new ways to glance at content on the wrist and produce information on the go through rapid microinteractions [6]. The primary input modality for the smartwatch is touch and hardware buttons, both of which require the second hand for operation. Motion gestures have been explored in research [5, 6, 61, 84]

Figure 5.1: (A) Synchro is a one-handed interaction technique for smartwatches that uses rhythmic correlation between the users's thumb and binary blinking controls. (B) A demonstration application with a blinking dismiss to voicemail icon (in blue). (C) Blinking targets can be placed in a vertical layout, such as in this reading application. (D) A music player with four possible targets. A wrist flick gesture is used to toggle between pairs of binary targets. (E) The intent to "sync" is detected when the thumb is repositioned (close to the hand) and icon color changes to indicate detection.

as another input modality, and have recently been adopted in commercial wearable devices for scrolling and selection.[1]

Synchronous gestures are based on the principle of rhythmic mimicry of their intent (e.g., movement or gaze) to a given stimulus, typically visual [17, 25, 112]. I motivate the use of a synchronous interface through an example with a common scenario in the home. A user, standing in front of a large TV display for gaming, holds a magic wand controller with motion tracking and buttons. The on-screen interface presents the user with multiple targets to select. If the user wanted to select one of multiple targets on this interface, there are a number of possibilities. One way is to assign a unique motion gesture of the wrist and controller per target. However, that would be inefficient and require the user to remember which gesture to use for each target. The user could use a few gestures (e.g., swipe left, swipe right, etc.) to navigate the interface and another gesture to select a target. The same operation could be performed using navigational and selection buttons on the controller, which is the common way of interacting today. In each of these cases, the user is expected to adapt to the interface. Synchronous gestures instead modify the interface to enable the

[1] https://developer.android.com/wear/preview/features/gestures.html

quick selection of multiple targets with a single gesture.

In this chapter, I present a synchronous gesture interaction for smartwatch input. The SynchroWatch technique is an example of a "moving controls" interface in which the user is expected to match their intent (e.g., thumb movement) to visual stimulus provided in the form of blinking selection icons on the smartwatch screen. In this work, I use blinking interface elements to represent possible sync selection targets. I use the smartwatch magnetometer to sense the continuous extension (away from the hand) and reposition (toward the hand) of the user's thumb, instrumented with a passive magnetic ring. The thumb movement approaches a sinusoidal waveform that is correlated to the blinking frequency of the target elements. The system supports binary selection by treating the turning points of the thumb extension and reposition as binary states (i.e., crest and trough of a sine wave) that correspond to the on-off states of a blinking target. There are a number of advantages to this approach: does not require users to memorize gestures, requires a small range of motion to interact, is intuitive, and has high discoverability.

The technique shifts the focus from two-handed techniques to one-handed smartwatch input. The system enables one-handed input by syncing the user's movement to visual targets as a convenient and subtle way of interacting with a small screen device. This work seeks to address three challenges of mobile input and gestural interfaces on smartwatches today: eliminating the need for calibration or training, minimizing false positive activation, and allowing operation while on-the-go. Introducing synchrony to an object blinking at a known frequency provides the user with feedback and also facilitates the first goal of requiring no user training. The algorithm auto-calibrates using the known blink frequency of targets and compensates for any latency in the user's stimulus response. Synchrony also supports the goal of minimizing false positive activation by reducing the search space of matching sensor data. Finally, I seek to empower the user to perform smartwatch input while they're engaging in daily activities, such as walking, leveraging movement correlation and expected movement patterns. I demonstrate the algorithm is robust and functions

while in motion. I evaluated the system through two evaluations. The first study is offline using sensor data collected with participants and analyzed post-hoc. Based on the findings from the first evaluation, I then implemented a prototype that runs in real-time on a smartwatch for the second study. Since the technique depends on relative and synchronous changes in the sensor data, it can be deployed on commodity watch platforms that are equipped with a magnetometer without knowing the exact location of the sensor inside the device. The second study runs in real-time on the device and provides users with feedback on their performance.

SynchroWatch gestures can be used to select between binary targets in various user scenarios. For example, as a *response* gesture, performed when a notification (visual, acoustic, or haptic) arrives on the watch. The user can quickly dismiss a phone call to voicemail by syncing their thumb movement to a blinking dismiss icon. SynchroWatch can also be used as a *command* gesture, within the context of an application, to select between binary targets. It is possible to extend the expressivity of the gesture using a complementary input modality, such as a wrist flick gesture. A flick gesture can toggle between different pairs of blinking controls, expanding selection beyond the basic case of two targets. Finally, SynchroWatch could with additional work be used as an *activation* gesture [95], with very low false positive and high true positive detection, to activate another set of command gestures or launch an application. However, the focus of the current system is for notification response and command gesture scenarios.

This work makes the following contributions:

- I describe an interaction technique for one-handed synchronous gestures on smartwatch devices that relies on synchrony between user movement and blinking on-screen controls.
- I describe a technique utilizing magnetic sensing with a passive thumb ring enabling one-handed, calibration-free, and user-independent gestures.

- I assess the technique using time-shifted correlation with a reference signal based on a simple physics model.
- I provide empirical evidence of the algorithmic performance through two user studies with eight participants in various distraction tasks (sitting, browsing on laptop, and walking).

5.3 SynchroWatch

The SynchroWatch technique leverages magnetic sensing with a thumb-worn ring. I chose to focus on magnetic sensing because I believe it enables a more robust sensing modality (versus accelerometer and gyroscope) for detection while moving. I describe the sync gesture, interaction design, interface and feedback design, hardware setup, and algorithms used in the technique in the following subsections.

5.3.1 Gesture Design

The synchronous gesture consists of a thumb extension and reposition. A visual representation is shown in Figure 5.2. I define a thumb reposition to be a "sync" action and a thumb extension as a "no sync" state. The gesture design is inspired by a classic mouse click and release. The reposition of the thumb, when closest to the hand, also provides tactile feedback with the thumb touching the side of the index finger. Intuitively, these two thumb states initially suggest the design of a binary controller with two on-screen targets. I describe additional possibilities on how to extend the interaction in the following section.

5.3.2 Interaction Design

The overall interaction possibilities for SynchroWatch span a range of possible approaches. Please see the demonstration applications section for more details.

SynchroWatch allows users to select an onscreen target by matching the motion of their thumb to the flashing of that onscreen target. SynchroWatch can be used with a single

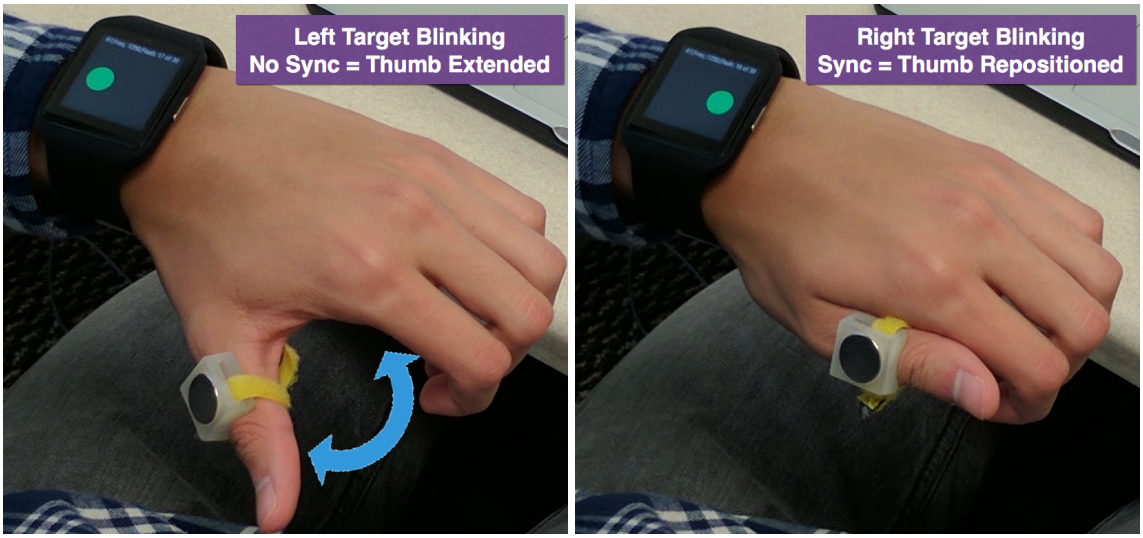


Figure 5.2: Visual representation of the SynchroWatch gesture, including a thumb extension (left) and reposition (right). A thumb reposition, closest to the hand, indicates a "select" or "action". In this figure, the user is syncing to the blinking target on the right side of the screen.

target on the screen, representing selection of a discrete command. This interaction style might be used in situations such as dismissing a text notification, or to return to the watch's home screen.

SynchroWatch can be used with two targets on the screen, allowing the user to select one of two discrete command alternatives. This interaction might be used in scenarios such as choosing whether to accept a phone call or send it to voice mail. SynchroWatch could potentially be expanded to allow selection from among more than two on-screen targets, but that is not in the scope of this work.

As envisioned interactions, Synchronous selection could also be combined with additional gestures to parameterize the action that is selected; this could be used to create "continuous" actions. For example, "sync and hold" allows the user, once a target is synced, to extend the duration of the selected command without the need to continue to sync. This could be used to continuously increase volume, or scroll position, based on how long the user's thumb is in the "hold" position.

Finally, the input expressiveness can also be extended through combinations with other

gestures. For example, SynchroWatch can be combined with gestures such as wrist flips to switch between modes or perform other commands.

5.3.3 User Interface and Feedback Design

The SynchroWatch user interface consists of two circular blinking targets of size 96x96 pixels each, a standard pixel resolution for smartwatch application icons. The blinking targets are flashing out of sync as seen in Figure 5.2. In this example, I focus on the target on the right side of the screen as the sync target. Figure 5.2 (left) shows only a target on the left side of the screen turned on, while the right blinking target which is the target of interest is turned off. In this figure, the thumb is extended in the "no sync" position. Figure 5.2 (right) shows the thumb in a "sync" position with the target of interest (circle on the right) turned on, indicating the user's intent to "sync" to the right target.

I evaluate three different frequencies (1.33 Hz, 1 Hz, 0.8 Hz) which correspond to different blink times (750 ms, 1000 ms, 1250 ms). To select a target, the user will "sync" to that target by moving their thumb, as explained in the previous subsection. The blinking circles are colored blue to indicate possible targets. Once the user has synced to a particular target, the blinking circle turns green to signify a sync gesture has been detected for that target. In practice, the size of the blinking circles targets can be smaller (10 x 10 pixels) than shown in the figure and placed alongside interface elements for integration into applications, reducing clutter in the interface. Alternatively, the on-screen interface elements (e.g., application icons, buttons, etc.) could be animated to blink or fade in/out at a known frequency.

5.3.4 Hardware Setup

I use a Sony Smartwatch 3 SWR50 for prototyping and evaluation. The device is based on a Quad ARM A7 1.2 GHz processor. I capture raw data from the smartwatch motion sensors, sampling the magnetometer, accelerometer, and gyroscope at the highest sampling

rates of 100 Hz, 200 Hz, and 200 Hz, respectively. The system samples the accelerometer and gyroscope to support *post hoc* analysis. I designed a 3D-printed mount for a magnet which sits on top of the user's thumb, and fastened the ring using a thin adjustable velcro strap. I use a rare earth Neodymium (N35) magnet disk with dimensions 16mm X 3mm for data collection and evaluation. The ring is designed as a prototype and can ultimately be fashionably designed to fit a user's lifestyle and thumb size.

5.3.5 Detector Pipeline

The SynchroWatch detector based on the algorithm and pipeline described below is designed and implemented as a Java library. The same detector can be used for both offline and online analysis. The detector runs on a laptop for offline analysis of magnetometer data collected during an experiment session (see Evaluation I described later). The detector is also incorporated as part of an Android application running on the Sony SmartWatch 3 and provides live feedback on syncing correlation (i.e., how well is the user syncing?) and direction (i.e., which binary target is the user syncing to?).

5.3.6 Algorithm Design

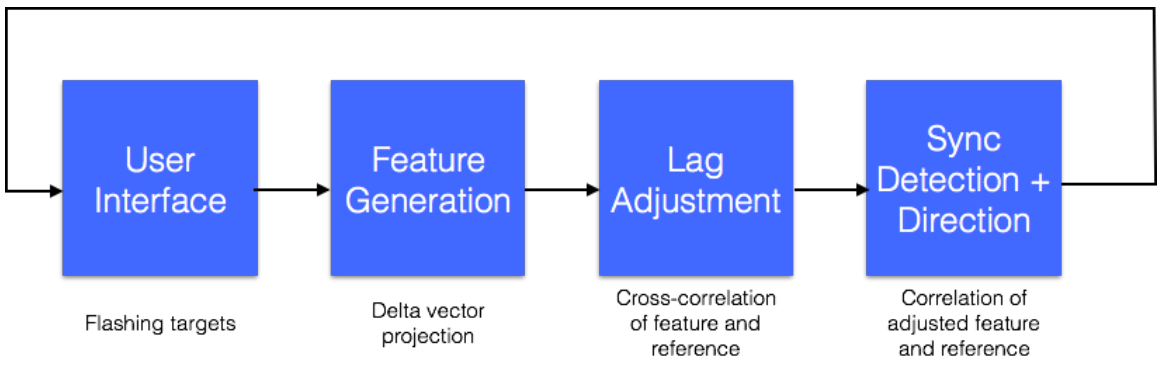


Figure 5.3: Visual flow of the SynchroWatch pipeline, including user interface, feature generation, lag adjustment using cross-correlation, and detection of syncing action and direction.

The pipeline consists of four main phases: user interface stimuli and feedback, feature generation using deltas vector projection, lag adjustment using cross-correlation, and the detection of syncing and direction. Figure 5.3 shows a high-level diagram flow of the pipeline.

Simple Harmonic Physics Model

I approximate the synchronous extension and reposition of the thumb at a given frequency as a simple harmonic motion. I ask the user to perform a periodic motion with the extension and reposition of the thumb. The resulting minimum-effort motion profile will loosely resemble a sinusoidal wave: an acceleration, then constant rate, then rapid deceleration. The waveform reaches its maximum point when the thumb is repositioned, and minimum point when extended. Figure 5.4 shows a 3D plot of the magnetometer vectors similar to a pendulum, while Figure 5.5 shows the sinusoidal pattern in the sensor data.

Magnetometer Sensor Data Pre-Processing

The magnetometer vectors are sampled at 100 Hz. The detector subsamples the vectors down to 10 Hz to remove signal noise and reduce the computation required for live processing on the smartwatch. The magnetometer vectors within every incoming 100 ms (i.e., 10 Hz) window are averaged. The timestamp for the averaged vectors is set to 50 ms (i.e., center of the window) away from the first magnetometer vector timestamp in the window. The timestamp of the left blink and right blink events are also passed to the detector.

Transformation Using Deltas Vector Projection

The motion of the thumb is captured by the magnetometer, each measurement resulting in a 3D sample. Let $H(k) \in \mathbb{R}^3$ be the magnetometer sample at time t_k. The magnitude of the measurement $M(k) \in \mathbb{R}$, given in (5.1), represents the magnetic field strength (with addition of noise) at each sample time. This magnitude is heavily influenced by the distance

of the magnet to the watch, as well by other surrounding magnetic fields.

$$M(k) \triangleq \left\| H(k) \right\|_2 = \sqrt{H(k)_x^2 + H(k)_y^2 + H(k)_z^2} \tag{5.1}$$

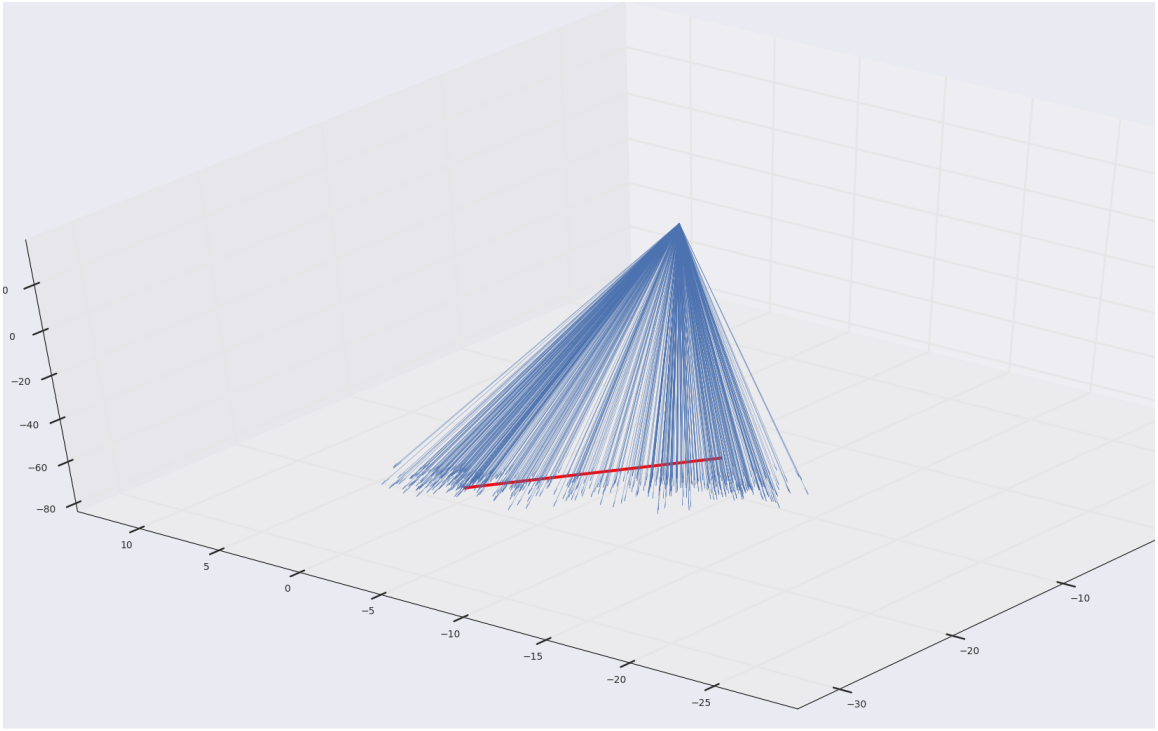


Figure 5.4: 3D visualization centered at (0,0,0) for the magnetic field vectors captured during a sync trial (1.33 Hz blink, browsing activity)

Intuitively, one might consider to use the magnitude of the magnetometer signal to sync with the reference signal (e.g., sine wave) as the natural thumb movements bring the magnet closer to the watch. However, based on pilot studies, this is not always the case: some results show a negligible variance in the magnitude, while the individual measurements in $H(k)$ exhibit a clear sinusoidal-like pattern (see Figure 5.5). In addition, no particular axis of the measurements (or a predefined linear combination) can be used exclusively, as the plane of the finger motion, and the ring magnet direction, may vary with the user.

I introduce a new feature that accounts for the dynamic and repetitive nature of the synchronous gesture. I define a 1D coordinate system based on the full sync reading cycle (i.e., left blink, right blink, left blink). The mean of all 3D magnetometer vectors sampled

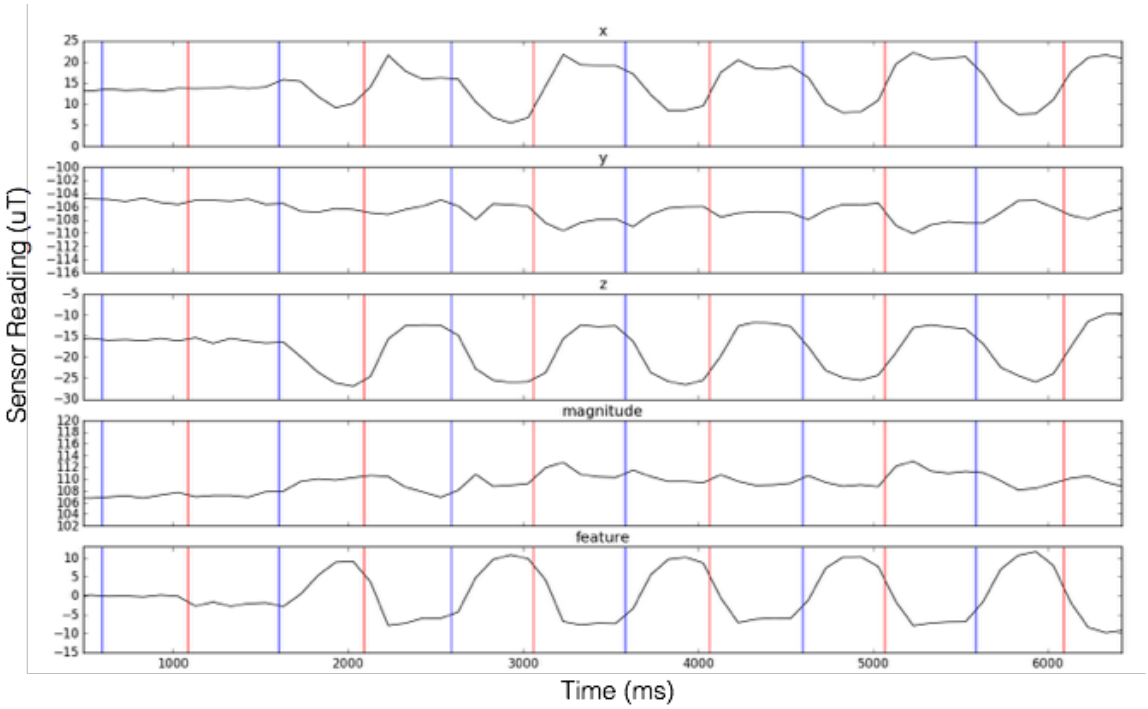


Figure 5.5: Visualization of raw magnetometer data for sync trial during sitting activity. The vertical lines indicate when the on-screen targets blinked. The red line corresponds to the left target and the blue line to the left target. The plotted lines correspond to the 3-dimensional components of the magnetometer: x, y, z. The magnetometer magnitude and delta vector projection are shown.

within the cycle are averaged and result in a magnetometer *mean vector* for the cycle. The two vectors farthest away from the mean are assigned to be the magnetometer vectors — *left vector* and *right vector* — that occurred at the first left blink and right blink in that cycle. The left and right vectors are subtracted resulting in a *basis vector* that is a vector line across the window. The vector of the current magnetometer reading (during a cycle) is subtracted from the mean vector to find the current *delta vector*. The current delta vector is projected onto the basis vector, resulting in a sinusoidal waveform, as shown in Figure 5.5. Overall, the vector projection using deltas significantly outperforms the magnitude across participants.

Once the system has computed the deltas, the data is detrended using a simple regression of all the points in the window to minimize effects of wrist motion and the earth's magnetic field changing around the user. The relative changes in the magnetic field in-

duced by the thumb movement are strong enough to preserve a measurable signal, even while walking. The detector interpolates the sensor data with a constant time interval to account for uneven sensor arrival times, a common known challenge of using a non real-time operating system with Android.

Determining Input Response Lag Using Cross-Correlation

The goal is to find the maximum correlation between the synthesized deltas feature and a reference waveform. The reference waveform is tuned to match the frequency of the blinking targets on screen, which can be adjusted as necessary.

When generating the reference waveform, it is necessary to account for input response lag to achieve the highest accuracy. This latency refers to the time delta between the visual stimulus and the thumb in "sync" state, and is caused by a variety of factors, most notably human reaction time caused by limitations in perception and motor response. Research has reported that human response times of haptic and visual stimuli is age-, gender-, and fatigue-dependent [78]. During pilot studies, I observed an average lag approximately 200 milliseconds between the arrival of the thumb at the sync position and the target blink. Thus, it is crucial to dynamically auto-calibrate and time-shift the reference waveform during usage, depending on the observed input lag, to ensure maximum correlation.

A window of 1.5 seconds, determined empirically, is used to cross-correlate between the feature and reference wave signal. The cross-correlation identifies similarity at different lags relative to one another. I find the optimal time shift to apply to the generated reference wave (see Figure 5.6).

After calculating the cross-correlation between the two signals, the maximum (or minimum if the signals are negatively correlated) indicates the point in time where the signals are best aligned, i.e. the time delay between the two signals is determined by the argument

of the maximum, or arg max of the cross-correlation, as:

$$\tau_{\text{delay}} = \arg\max_{t}((f \star g)(t)) \tag{5.2}$$

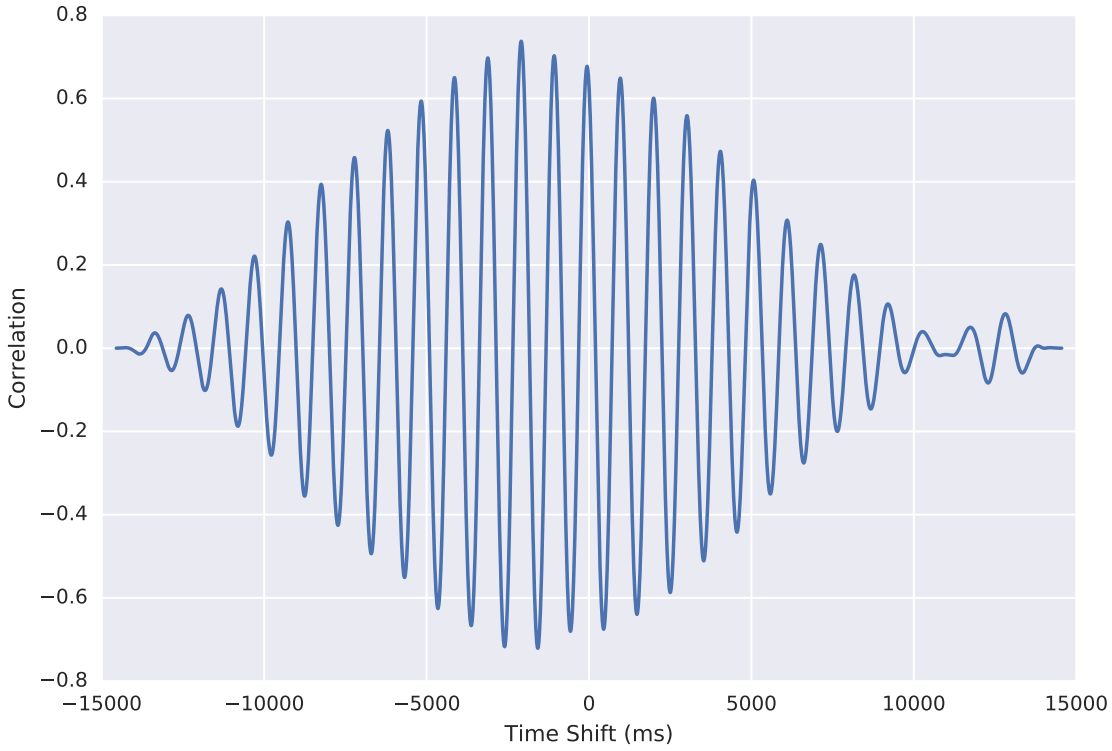


Figure 5.6: Output of the cross-correlation between a reference sinusoidal waveform and the magnetometer signal sampled from the watch. Cross-correlation shown here is calculated over a 15 second window resulting in a two second lag. The time delta between zero and the location of the argmax determines the suggested time shift for a particular trial.

Time-Shifted Sine Wave Correlation Using Deltas

Once the time-shifted reference wave is aligned with the sensor data, the detector performs the correlation between the two signals to detect the synchronous gesture and target. I use correlation as a statistical measure of dependence between the two sets of data.

I leverage the commonly used Pearson product-moment correlation coefficient, $\rho_{X,Y}$, which measures the linear dependence between two variables X and Y. See Equation 5.3.

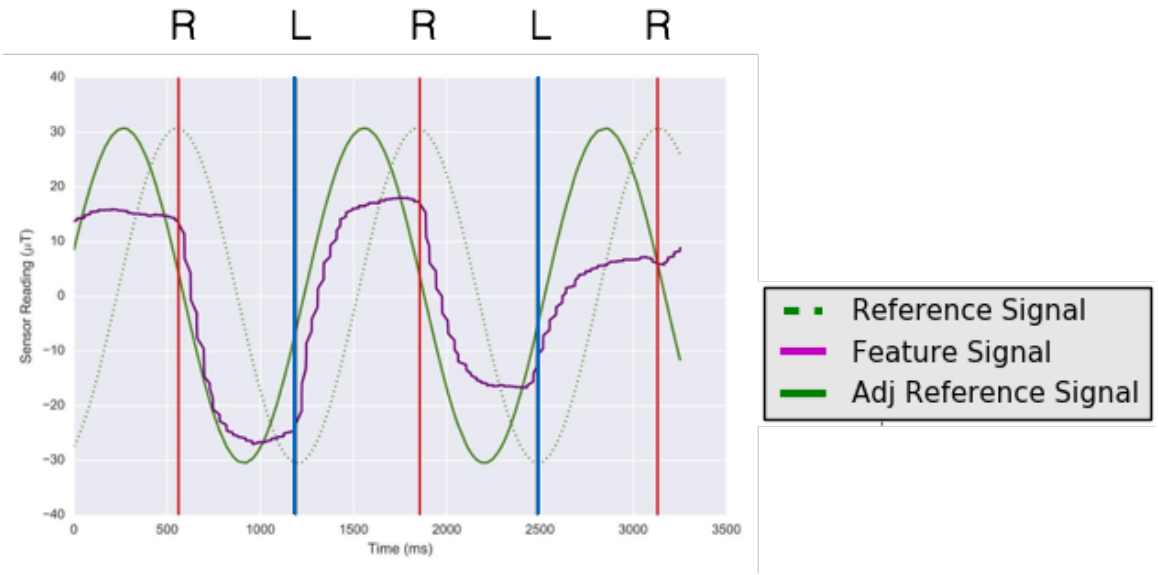


Figure 5.7: A short window of data highlighting the delta projection feature overlay with the reference sinusoidal waveform. Frequency is 0.8 Hz or period of 1250 ms. The dotted green line indicated the reference sinusoid signal centered around the flashes (i.e., red line is a right target flash, blue line is a left target flash). The purple line represents the feature based on the user's thumb movement and appears early. The solid green line is the reference sinusoid signal after time-shifting using cross-correlation.

The total correlation between the two variables is measured as a value from -1 to 1, where $\rho_{X,Y} = 1$ denotes total positive correlation, $\rho_{X,Y} = 0$ is no correlation, and $\rho_{X,Y} = -1$ is total negative correlation.

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{5.3}$$

The correlation coefficient is used to gauge the similarity of the sensor readings and the reference wave and to determine whether the user input is synchronized with either one of the on-screen stimuli. Figure 5.7 shows a small window of the delta vector projection with the overlay of a sinusoidal reference waveform. The system is able to detect (a) whether a "sync" event is being performed by exceeding a correlation threshold and (b) which blinking element is being targeted based on the directionality of the coefficient (± 1). Figure 5.8 shows an example of a user syncing to the right target at 1 Hz.

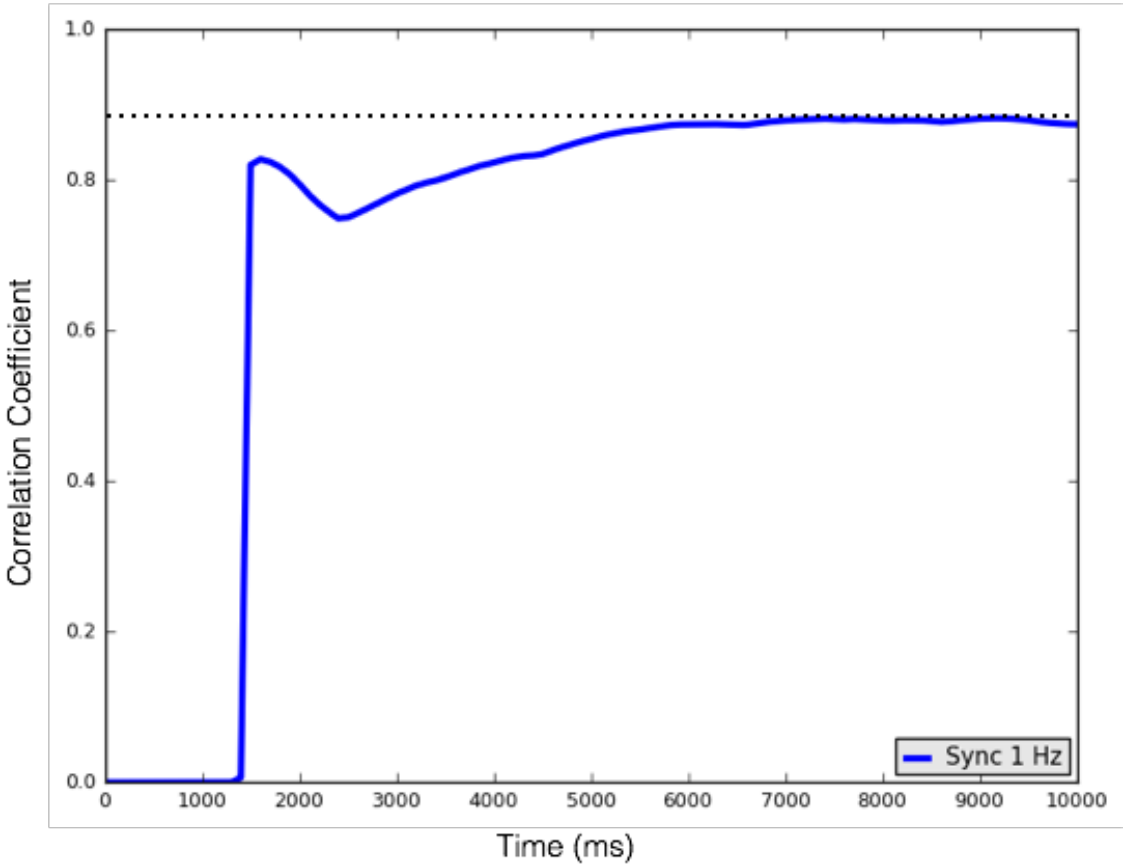


Figure 5.8: Example figure of the correlation over time graph for a user syncing to the right target at 1 Hz.

5.4 Evaluation I - Offline Assessment of SynchroWatch Syncing Correlation

I conducted a technical and user evaluation of the interaction technique with an unmodified Sony Smartwatch S3 in the usability lab of our institution. Upon arrival, participants signed a consent form. After the study, participants completed a questionnaire to assess their experience during the study. The study lasted approximately one hour per participant.

5.4.1 Participants

I recruited eight participants from our institution (6 male, 2 female, ages 18-30) via word of mouth. All participants are students. Seven participants indicated not wearing a smartwatch

regularly. Participants received $10 compensation for their time.

5.4.2 Design and Experimental Setup

Participants wore the watch and magnetic ring on their left hand. To begin, researchers performed a demonstration of the gesture to be performed. Participants familiarized themselves with the data collection application during a practice round.

The study was designed to capture sensor data in a semi-controlled environment. Participants were asked to perform the following tasks naturally and were not given any special instructions. During the evaluation, they were prompted using haptic feedback on the smartwatch, using a random Poisson delivery time of one minute, to perform a sync event.

When prompted to perform the sync gesture, they raised their arm and watch to line of sight and completed the sync. For the purposes of the evaluation, all participants were asked to sync to the right on-screen target. The targets were flashed at different frequencies (1.33 Hz, 1 Hz, 0.8 Hz) which correspond to different blink times (750 ms, 1000 ms, 1250 ms) for 15 cycles after the prompt. I consider a full cycle to mean the flashing of the left and right targets. The size of the two targets was 96 x 96 pixels, each centered on the left and right half of the screen.

The three distraction tasks (see Figure 5.9) assigned to participants were:

- Browsing: The participants sat at an office desk in front of a laptop. They were asked to search online and type out the responses to a list of 20 random trivia questions (e.g., What is the capital of Denmark?, What is the unemployment rate in your country?, What is 33% of 23,857?, How many inches are in 5 miles?, etc.). The task was designed to simulate every day use of a laptop computer including using the trackpad, typing text, and pressing number keys.

- Walking: Participants walked along an indoor walking path. The path included walking in straight lines and random turns. At least one researcher walked alongside participants and engaged them in conversation, while the task was taking place.

- Relaxing: The participants sat in front of a laptop and were given the option to choose their favorite sitcom or movie to watch. Participants were asked to sit naturally and rested their hands on their lap for the duration of the exercise while the video played.

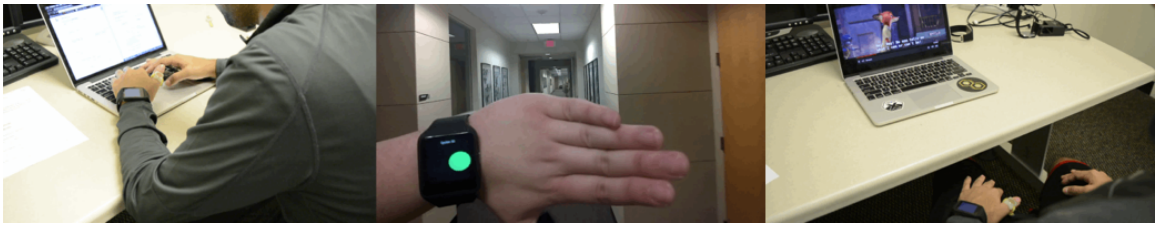


Figure 5.9: User study distraction tasks: browsing on the computer, walking, and relaxing watching videos. The hand was raised to chest level during the evaluation. For demonstration only, the hand is shown raised to eye level in the walking condition.

Data was recorded for the entire session from beginning to end of each distraction task, which included approximately 4-5 minutes of sync data and 15 minutes of noise data (in which no intentional sync events were performed) for each of three activities. In total, I collected 1 hour of sensor data for each of the 8 participants. During each activity, the system prompted participants 5 times for each frequency for a total of 15 rounds. The order of the tasks and the blink frequencies were randomized for each participant. In summary, the user study includes 8 participants x 3 distraction tasks x 3 frequencies x 5 repetitions per frequency x 15 sync cycles per round for a total of 5400 sync cycles.

5.4.3 Results

The steps of the detection algorithm include: the calculation of the deltas vector projection feature, determining input lag using cross-correlation with a sinusoidal waveform, and correlation between the two signals to determine a similarity coefficient. I present results for each of these phases and some additional analyses.

Determining Input Response Lag

I visually demonstrate that the delta vector projection can outperform magnitude (see Figure 5.5) in the first stage of the algorithm. Here, I present results for the second stage determining the input lag between the user's response and visual stimuli using cross-correlation. On average across participants and all 360 sync trials (8 participants x 3 activites x 3 frequencies x 5 rounds), human reaction time was approximately 429 milliseconds. See Figure 5.10 for the breakdown by frequency.

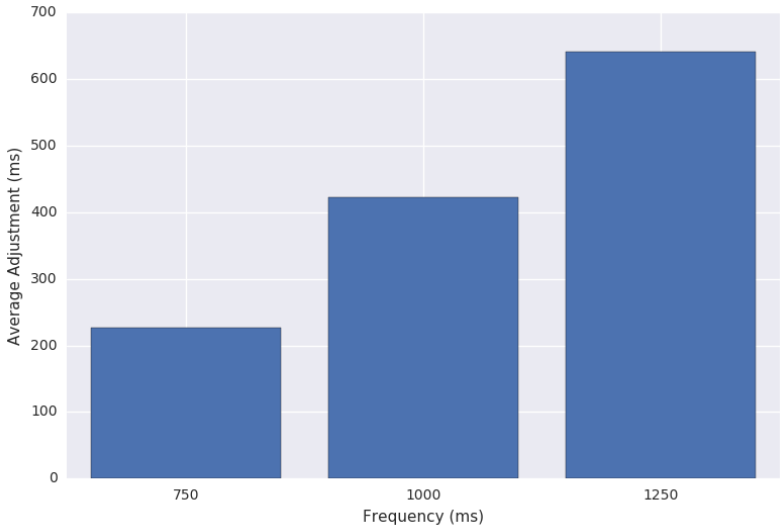


Figure 5.10: Average response lag across all participants.

Correlation Over Time

I use the input response lag time computed per trial to auto-calibrate and adjust the reference waveform. I then calculate the correlation over time to the deltas vector projection (as explained in the algorithm section). The correlation over time is calculated by running the detection technique on growing windows of data from 0.1 sec in 0.1 sec intervals until the full 15 cycles of data are included. I achieve an average correlation coefficient between 0.5 and 0.6 between the participant's gesture and the sync signal (see Figure 5.11).

I also compute the correlation over time using the delta vector projection against the

noise or null dataset. Specifically, I chose 100 random samples from the null class of the appropriate window size, again growing the window size from 0.1 sec to full 15 cycles of data in 0.1 sec intervals. I achieve an average correlation coefficient between 0.1 and 0.2 for the null class data, which is significantly lower than the correlation when the user is synchronizing. As seen in Figure 5.11, the correlation coefficients between synchronized gestures and normal movement become well separated within three seconds, suggesting that a detector can be made that detects a synchronized gesture relatively quickly.

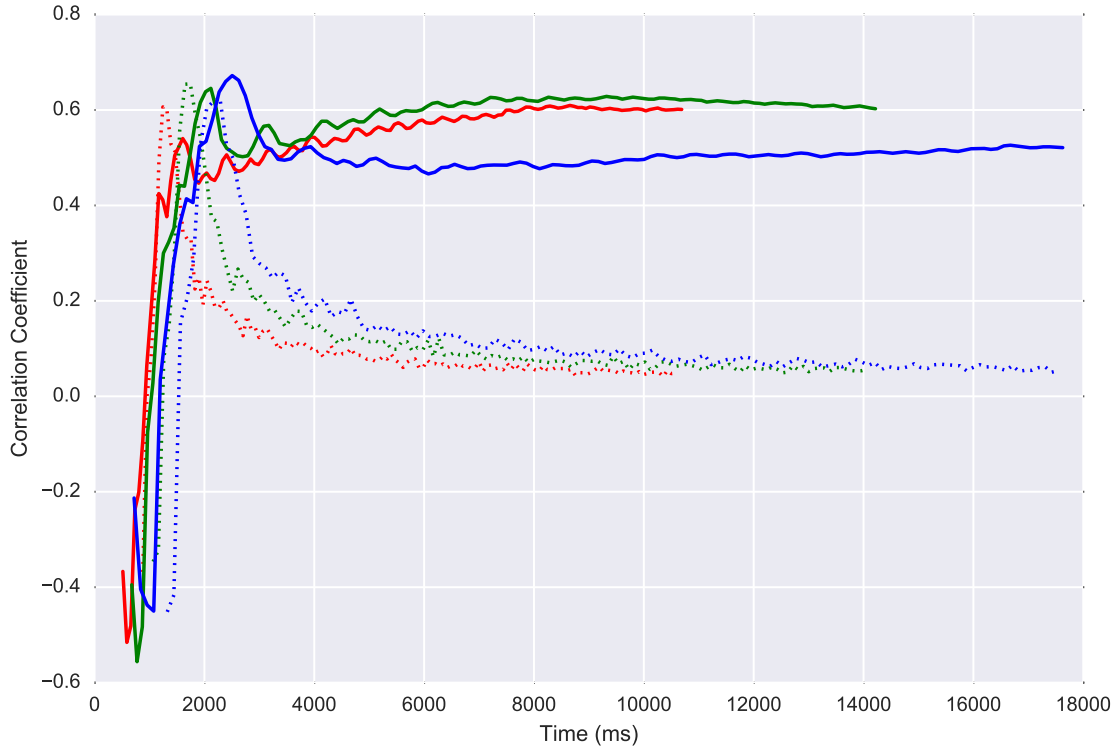


Figure 5.11: Correlation over time using delta vector projection for signal (solid lines) and noise data (dotted lines). The colors indicate the blinking frequency (blue = 1.33 Hz, green = 1 Hz, red = 0.8 Hz). The correlation is calculated over 15 cycles for different frequencies, explaining the different line lengths.

Qualitative Feedback

After experiencing the technique during the evaluation, I asked participants to rate the system on a number of dimensions. Participants were asked to rank four attributes (comfort

of use, ease of learning, usefulness, and overall experience) on a 7-point Likert scale, 1 being the lowest and 7 being the highest score. Overall, participants scored the attributes as follows: comfort of using the system (μ=4.88, σ=0.83), ease of learning (μ=6.00, σ=0.76), usefulness of the system (μ=5.13, σ=0.99), and overall experience using the system (μ=5.13, σ=0.83).

I also asked participants to comment on what they liked about the system, suggestions for improvements, and what they would like to use SynchroWatch for. Most users reported the ability to interact one-handed and the low learning curve as the best part about Synchro. P7 said it was "Easy to learn. Felt like I'm playing a little game every time." P5 reported "I liked that I didn't need to do anything with my other hand. I just had to move one finger which was very easy." P3 was happy about the "simplicity of the use and integration with daily activities." P1 similarly thought "I like that it is a simple gesture that could be applied in a lot of ways."

I used a 3D-printed ring with an adjustable strap to fit multiple hand sizes. In the future, I believe a magnetic ring customized to fit the particular user's hands would be most comfortable. Some users reported slight discomfort due to the fit of the ring on their thumb. A few users were unable to consistently sync with 1.33 Hz blinking while walking. P7 said "Walking was hard at certain frequencies. Maybe have the user set a comfortable speed for themselves?" P4 suggested "Have different frequencies for different actions." P2, P4 and P7 also proposed the use of haptic feedback along with visual flashing to allow simple eyes-free interactions.

Finally, users also suggested multiple applications for Synchro, such as responding to phone calls, selecting items for quick short typing or canned responses, playing games like Flappy Bird, scrolling through a text message or email, and playing/pausing a song. I leveraged user feedback and early explorations to motivate the implementation of three demonstration applications described below.

5.4.4 Discussion

I generate the correlation over time graphs between the sensor readings and a reference sine wave to determine the length of time needed to adequately detect a user syncing. In general, the goal is to find the least amount of time needed to converge past a detection boundary. Based on analysis, I observed that an increased window length leads to higher correlation with an asymptotically increasing pattern. Initially, the correlation calculation is seemingly unstable with the presence of a peak prior to stabilizing. I hypothesize the peak is likely due to participants moving their wrist in synchrony with the stimulus when randomly prompted during data collection. This initial wrist movement of raising the watch to the field of view is visible in the raw magnetometer data (see Figure 5.5) and causes unpredictable results during the initial seconds of detection. A way to mitigate this behavior is to set appropriate correlation thresholds and empirically set a waiting time before triggering.

QuickDraw by Ashbrook et al. [7] investigated the effect of placement and user mobility on the time required to access an on-body interface. The researchers found that a wrist-mounted system was significantly faster to access than a device stored in the pocket or mounted on the hip and reported access time of 2.787 sec. Access time includes reaching for the watch and responding to an alert. While the repetitive nature of the SynchroWatch gesture may appear slow, the technique is only slightly slower than accessing the touchscreen with the second hand. It takes approximately 3 seconds to achieve a stable correlation coefficient above 0.5 for the three frequencies, after raising the arm and beginning to sync.

One of the advantages of the SynchroWatch technique is that it automatically compensates for the user input response lag and does not require calibration. In addition to the evaluation using a Sony Smartwatch 3 device, I also successfully deployed and ran the algorithm on an LG G Watch device. The technique is agnostic to the orientation of the sensor and the sensor topology inside the smartwatch, as it leverages relative changes in the magnetic field across axes with the delta vector projection feature.

5.5 Evaluation II - Live Comparison of SynchroWatch vs. Touch

The goal of the first study in Section 5.4 was to evaluate participants' offline performance syncing at three different frequencies (i.e., 0.8 Hz, 1 Hz, 1.25 Hz) during three different activities (e.g., browsing, walking, relaxing). The goal of the second study is to evaluate the online performance of participants using two interactions — syncing vs. touch swiping. The systems in this study use sensor data from the device for online analysis and live feedback is provided for syncing and swiping. Sensor data is recorded for both the swiping application (e.g., touch location, gyroscope, accelerometer, magnetometer) and the syncing application (e.g., accelerometer, magnetometer).

5.5.1 Syncing vs. Touch Swiping

During the evaluation, participants were asked to imagine they were in a meeting and received a visual/haptic notification on the smartwatch. Their task was to dismiss the notification as quickly as possible using two distinct interactions — syncing and touch swiping.

The first interaction to dismiss a notification in this evaluation used SynchroWatch by "syncing right", which only requires the use of the watch hand and is the focus of this work. A correlation threshold trigger of 0.9 is used to trigger syncing. The color of sync target changes color (blue to green) based on the correlation score and sync direction. A higher correlation results in the syncing target turning green. Once a trigger above the threshold was detected, a text display indicating a sync detected was shown.

The second interaction was touch "swiping right" using the second hand to swipe a target on the left side of the screen to the right side of the screen. Once the icon being swiped had reached the other side of the screen and was within the touch buffer zone, the target changed colors to green as feedback of task completion to the user. A text display indicated a swipe detected was shown once the participant released the target inside the target zone. This action follows the current mode of interaction on many commercial watches

and phones today to slide to unlock the device or answer a phone call.

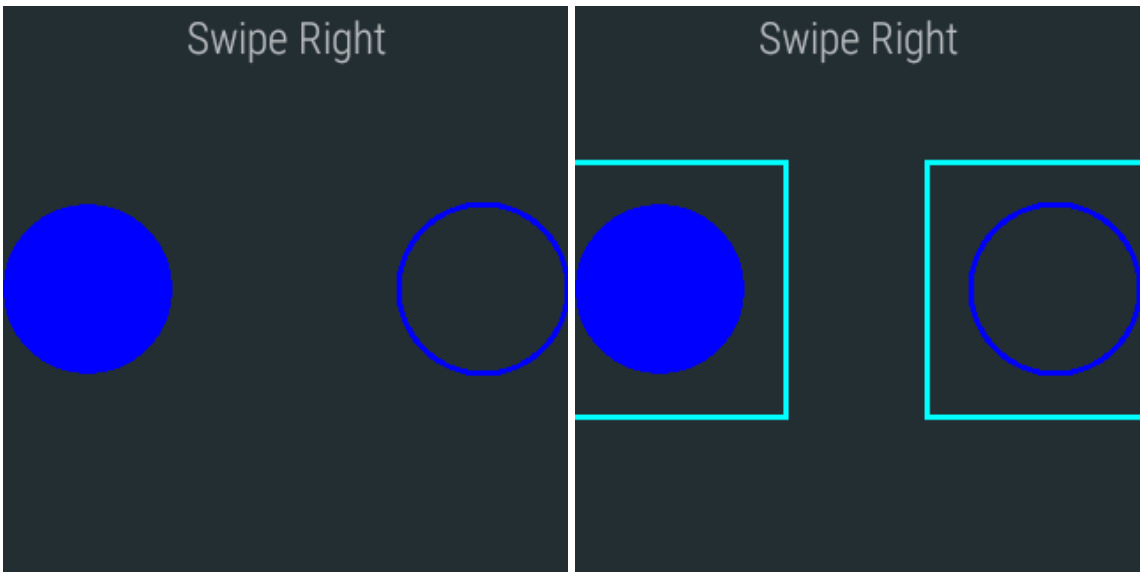


Figure 5.12: A) Screenshot of the swipe application interface presented to the user with two selection targets/icons. B) Swipe interface showing relaxed touch target areas for selecting the right or left targets to swipe.

5.5.2 Participants

I recruited eight participants from our institution (5 male, 3 female, ages 18-28). All participants are students. The study lasted approximately two hours. Participants received $10 compensation for their time. The majority of participants reported not regularly wearing a smartwatch or other wearable device. Three participants did indicate multiple years of musical experience (e.g., playing an instrument or reading music), which may facilitate "syncing" to a particular beat or stimulus.

5.5.3 Design and Experimental Setup

I conducted a technical evaluation of the interaction techniques (sync vs. swipe) with an unmodified Sony SmartWatch 3 in the usability lab of our institution. Upon arrival, participants signed a consent form. After the study, participants completed a questionnaire to assess their experience during the study.

The within-subjects study was designed for two conditions (sitting and walking) and two interactions (syncing and touch swiping). A 4x4 Latin square design was used to counterbalance the sitting/walking and syncing/swiping. Two practice sessions were completed, followed by two evaluation sessions. Participants were randomly assigned to a row of the Latin square for the first practice session. The same order of the Latin square was used for the second session. A new randomly assigned row of the Latin square was used for the first evaluation. The same order of the square was used for the second evaluation.

Participants wore the smartwatch and magnetic ring on the left hand. To begin, researchers played an introduction and training video to participants. The video asked participants to imagine they are in a meeting and are receiving a phone call through a visual/haptic notification. They are asked to dismiss the notification using syncing or swiping. Participants familiarized themselves with the data collection applications for syncing and swiping by performing (while sitting) at least 5 sync examples and 5 swipe examples.

The study was designed to capture sensor data in a semi-controlled environment. During the study and for both interactions, the participant was asked to keep their arm and hands on their lap (while sitting) and down by the side of their hip (while walking). Participants were asked to perform the following tasks naturally and were not given any special instructions. For both applications, a haptic vibration was delivered to the user randomly using a Poisson delivery time of 10 seconds prompting them to raise their arm to chest level and line of sight. The interface controls for syncing and swiping appeared on-screen as soon as the device vibrated.

For the purposes of the evaluation, all participants were asked to sync and swipe to the right on-screen targets. For syncing, the targets were flashed at 1 Hz corresponding to a blink period of 1000 ms. I consider a full cycle to mean the flashing of the left and right targets. A correlation threshold of 0.9 was used to trigger a sync event. The application prompted users to sync for 10 cycles after the prompt, as well as swipe right 10 times. The size of the two targets was 96 x 96 pixels, each centered vertically on the far left and right

half of the screen.

The two distraction tasks assigned to participants were:

- Walking: Participants walked along a random indoor walking path. The path included walking in straight lines and random turns. At least one researcher walked alongside participants and engaged them in conversation, while the task was taking place.

- Sitting: The participants sat in front of a laptop and were given the option to choose their favorite sitcom or movie to watch. Participants were asked to sit naturally and rested their hands on their lap for the duration of the exercise while the video played.

Data was recorded in individual files during each of the sync and swipe tasks in each of the distraction tasks. The length of a syncing session with 10 syncs was approximately 3.5 minutes. The length of a swiping session with 10 swipes was approximately 2.5 minutes. Noise data was collected for the interim period when the participants are waiting to be prompted to perform the task.

In total, I collected 2 hours of sensor data for each of the 8 participants across two practice sessions and two evaluation sessions for sitting and walking.

For each sync session, the system prompted participants 10 times to sync. In summary, the user study included 8 participants x 2 distraction tasks x 1 frequency x 10 sync repetitions per frequency x 10 sync cycles per round for a subtotal of 1600 sync cycles. Across two practice and two evaluation sessions, I collected a total of 640 sync events.

For each swipe session, the system prompted participants 10 times to swipe. In summary, the user study included 8 participants x 2 distraction tasks x 10 swipe repetitions for a subtotal of 160 swipe events. Across two practice and two evaluation sessions, I collected a total of 640 sync events.

5.5.4 Results

Syncing - Correlation Over Time

The correlation over time is calculated live by running through the detection pipeline described above. A correlation coefficient is output every 100 ms for a 1.5 second running window of sensor data. The correlation coefficient increases as a function of time reaching between 0.8 and 0.9 for the syncing event (see Figure 5.13). Also shown are normalized values for the acceleration vector including gravity. The acceleration data shows the movement of the arm raise from a rest state to syncing state. The time between intention and action can be described as the time it takes the user to raise the arm to provide input, which can be inferred from this accelerometer data.

Syncing - Accuracy vs. Time vs. Thresholds

To further analyze whether the synchronized gesture might be detected accurately versus normal movement, I plot a 3D graph of correlation coefficient vs. time vs. accuracy. Using different thresholds, I run the detector post-hoc on the data where the participants are trying to synchronize their thumb movements to the graphics and the interim period where they are just sitting or walking. The results are shown in Figure 5.14. Accuracy is defined in this graph as the average between the percent accuracies for detecting true positives and true negatives. True negatives are defined as 1 - false positive rate. The optimal results for sitting and walking indicates syncing is possible within 4 seconds at a correlation threshold of approximately 0.7.

Syncing and Swiping - Time To Trigger

The time between intention and action (see above) and the interaction time have been previously suggested as useful metrics to assess the value of an interaction [103]. Microinteractions are quick actions that typically last only a few seconds [6].

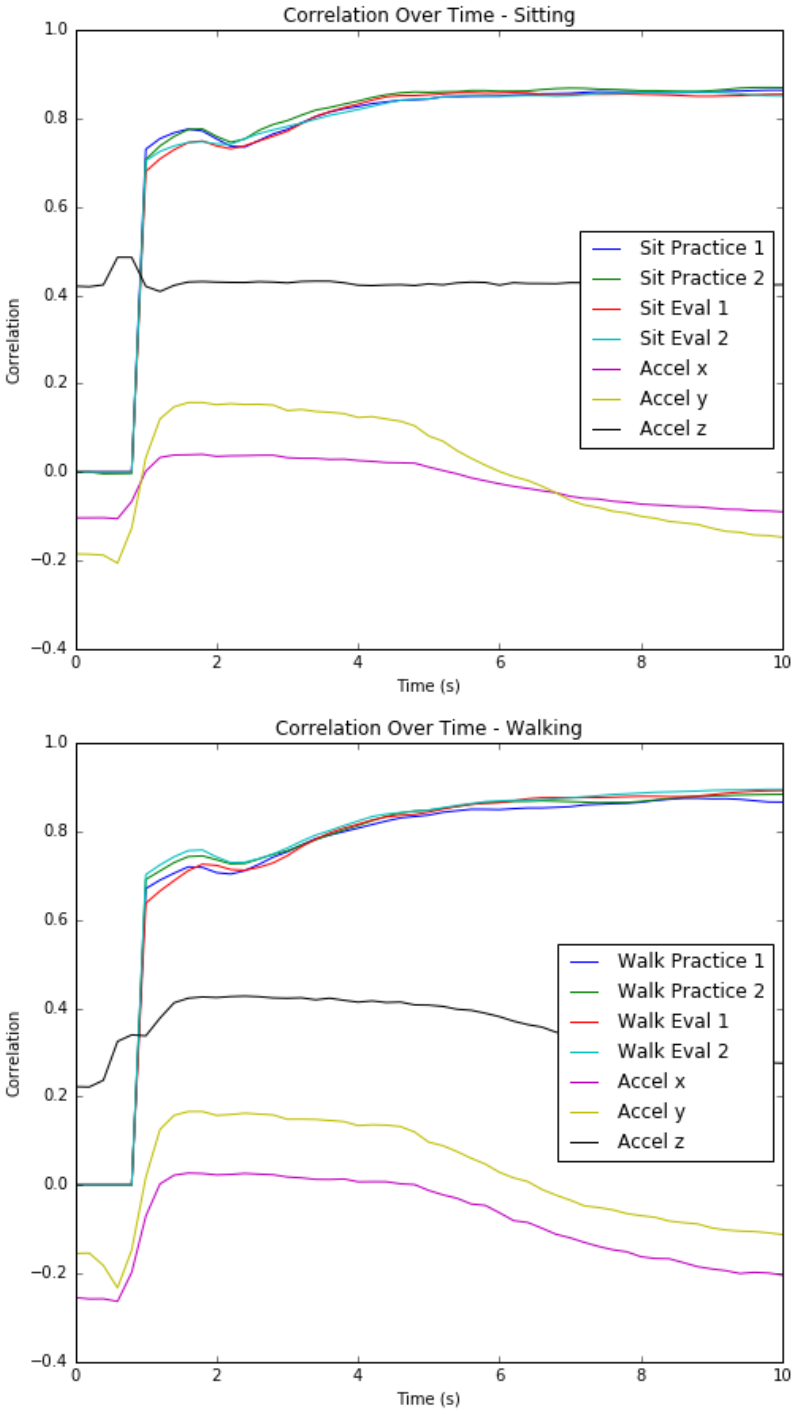


Figure 5.13: Correlation over time using delta vector projection for sitting and walking. Figure also includes the normalized acceleration vector of the wrist including gravity showing when the arm is raised for syncing (within the first 1.5 to 2 seconds).

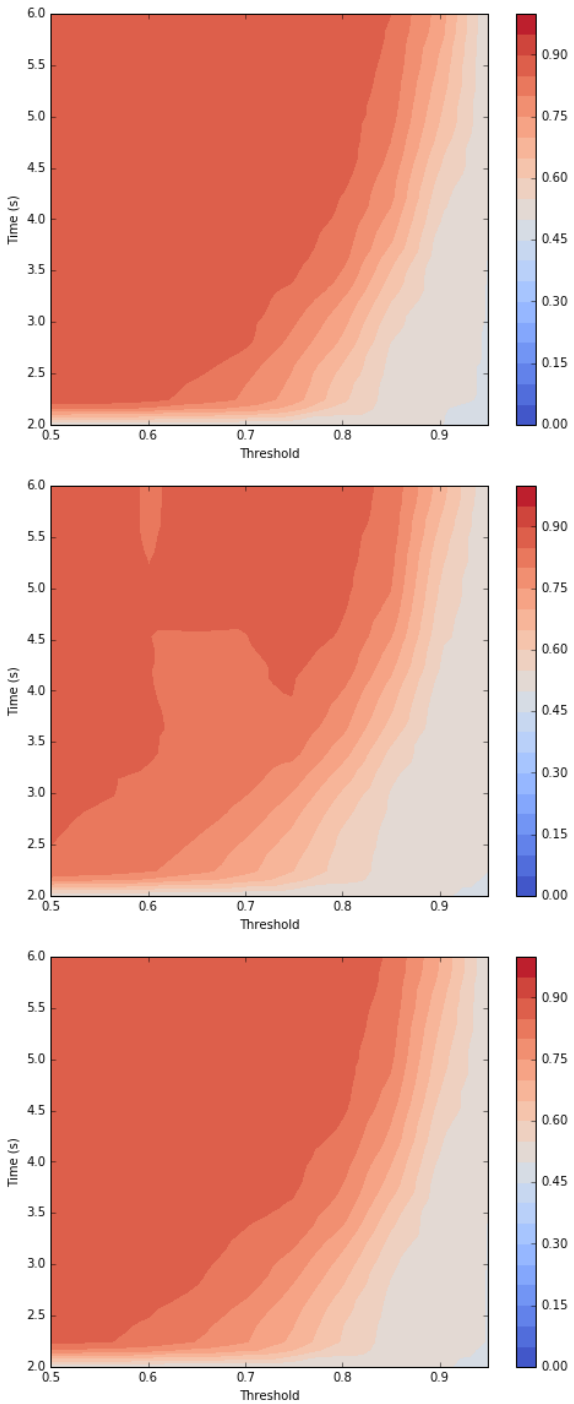


Figure 5.14: 3D plot of threshold vs. time vs. accuracy over time for sitting, walking, and sitting and walking combined.

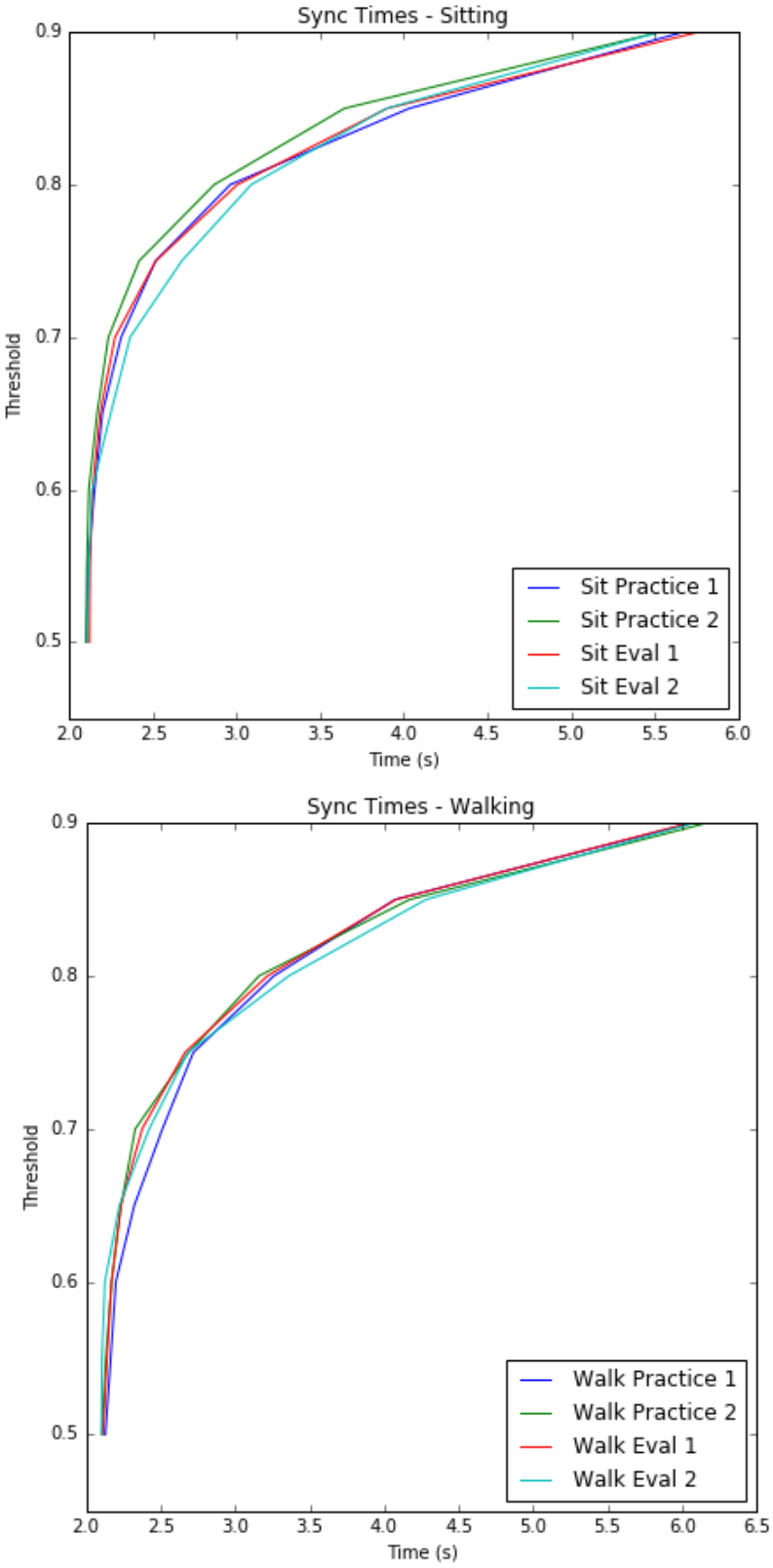


Figure 5.15: Figure showing time to sync for A) sitting and B) walking at various correlation thresholds.

The time to completion or is one metric of interest I evaluate for both interactions (i.e., syncing and swiping). For syncing, the time to sync depends on the correlation coefficient threshold used to trigger. During the live evaluation, a correlation coefficient of 0.9 is used as the trigger threshold. A 0.9 coefficient was observed on the upper end of the syncing scores during pilot testing. Additionally, a conservative threshold results in higher times to trigger. However, it also enables further analysis post-hoc by reducing the threshold to be less conservative and reassessing the time to trigger.

The time to trigger includes the time required by the user to raise the arm from rest to chest level and complete the swipe. Thus, the total time to trigger includes the time between intention and action (i.e., setup time) and the interaction time.

As observed in the correlation over time graphs (see Figure 5.13), no correlation coefficient is reported for the first window of data. The earliest time the system could potentially trigger is set to 1.5 seconds to avoid triggering while the person is raising the arm.

Overall, it took participants approximately 5.5 seconds to complete the sync event on average with a correlation coefficient of 0.9. By reducing the coefficient to 0.8, I observe that sync times can be reduced to under 4 seconds. For swiping, the time to trigger was on average 2.1 seconds across both sitting and walking. The time to trigger includes the time required by the user to raise the arm from rest to chest level and complete the swipe. See Figure 5.15 and 5.16.

Task Workload

The NASA-TLX questionnaire measures subjective workload ratings. Previous studies have indicated that it is a reliable and valid measure of the workload imposed by a task [15, 43, 46]. As Lyons et al. indicate based on their work on evaluating text entry rates on a Twiddler device [70], subjective workload ratings can be more sensitive to working memory demands after performing the task than are measures of performance. In addition, subjective ratings can be informative when a task is difficult yet within the individual's

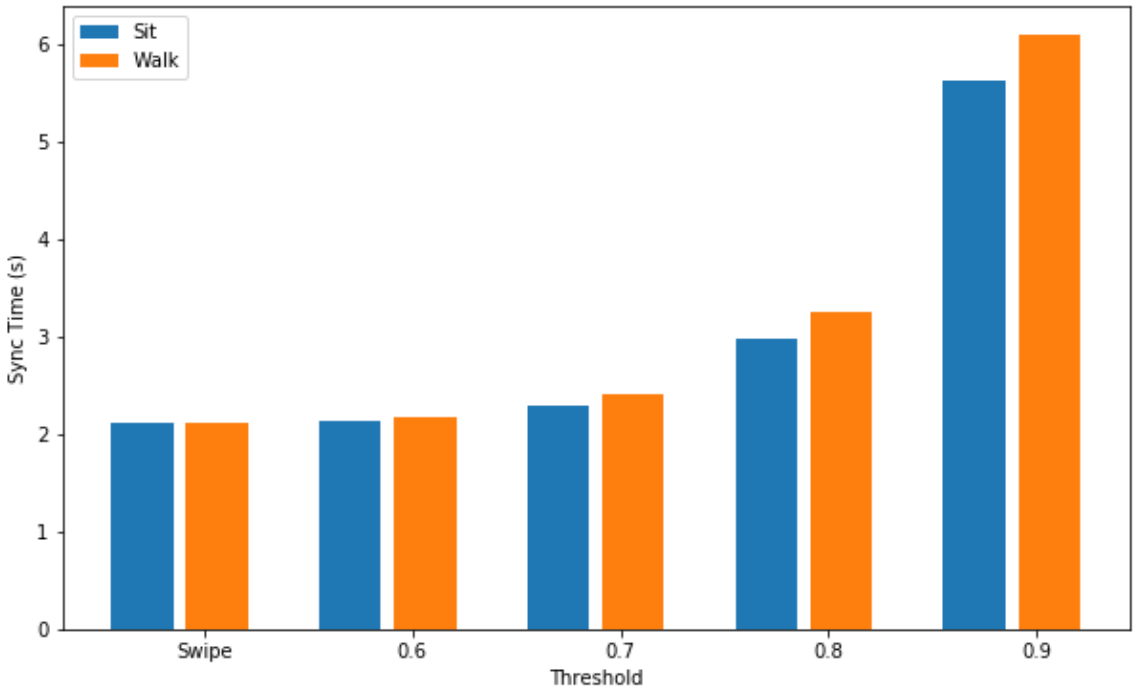


Figure 5.16: Summary of the times to trigger for swipe/sync and sitting/walking. Sync times are reported for various correlation thresholds (0.6 to 0.9).

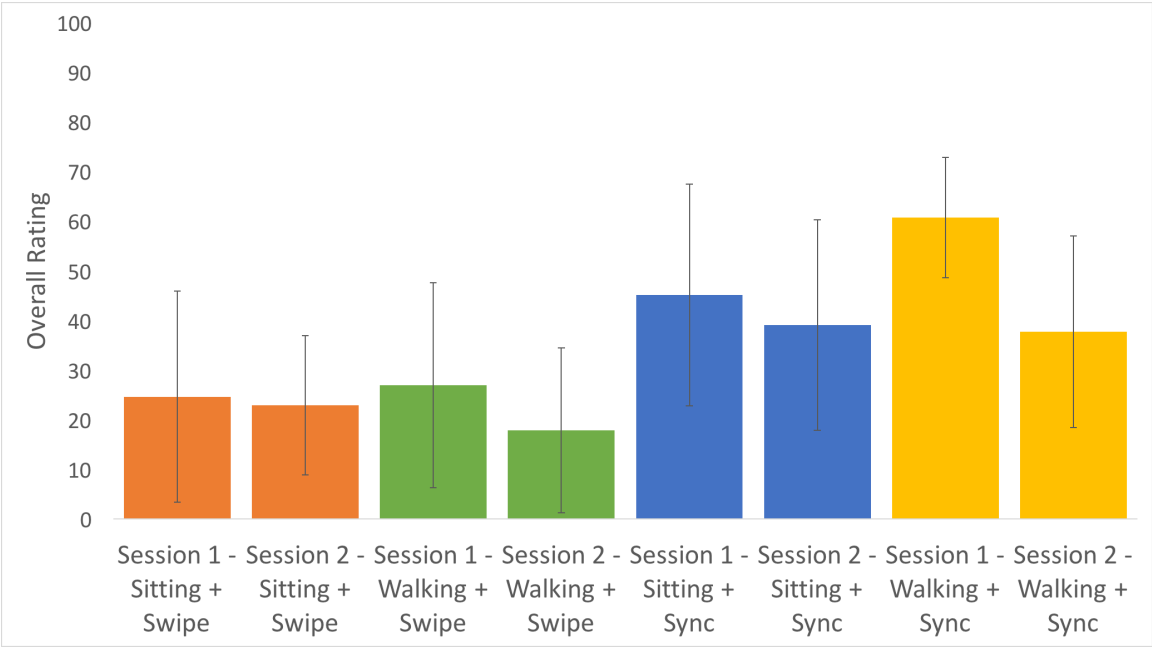


Figure 5.17: Bar graph showing the overall workload for each condition in the evaluation phase of the study for sitting/walking and syncing/swiping.

capability. For instance, as a task becomes more difficult, the individual can increase his or her effort to maintain the same level of performance. In this case, subjective ratings of workload could capture this increased effort, whereas performance measures could not [70, 119].

The NASA-TLX consists of six scales: mental demand, physical demand, temporal demand, performance, effort, and frustration; each scale has 21 gradations. For each scale, individuals rate the demands imposed by the task. In addition, they rank each scale's contribution to the total workload by completing 15 pairwise comparisons between each combination of scales. This procedure allows an investigation of the task-demands load on each scale, as well as a measure of the overall workload. Interpretation of the mental, physical, and temporal demand scales are straightforward; each scale captures the demand imposed by its title. The performance scale captures how successful participants felt they were at accomplishing the given task. The effort scale captures how hard individuals had to work to achieve their level of performance; both mental and physical effort can contribute to this scale. The frustration scale captures how much the task annoyed or discouraged the participants. The overall workload rating is calculated by summing the product of each scale's rating and weight. This calculation results in a score between 0 and 100. It reflects an individual's perception of the amount of workload devoted to each of the scales, along with each scale's contribution to overall workload. Further description of NASA-TLX is described by Hart et al. [43].

For the study, I analyzed the overall workload ratings in addition to the six individual scale ratings. Participants completed the NASA-TLX questionnaire after each condition (sitting-swipe, sitting-sync, walking-swipe, walking-sync) in each of the two evaluation sessions of the study, for a total of 8 completions. Participants entered their ratings on a tablet-based NASA-TLX application. The application first asked the participants to rank each of the 6 scales and then randomly provided 15 pairwise comparisons between each combination of the scales. Caption text to describe each scale is provided.

The overall workload for each of the evaluation sessions averaged across participants is shown in Figure 5.17. The task with the highest workload is the first session of walking and syncing at 60.8 out of 100. The second highest workload corresponds to the first session of sitting and syncing at 45.1. The lowest overall workload is measured for swiping while walking after two practice sessions and one evaluation session. Standard deviations suggest no significance between the conditions. Although syncing does present a higher perceived workload overall, it is encouraging to observe that the average perceived workload decreases for sitting/walking for the second evaluation session, after two practice sessions and one evaluation. Swiping is a common action performed on touchscreen-enabled devices today. It is possible that as users become more and more familiar with the syncing technique that the perceived workload would continue to decrease and muscle memory may improve the overall experience and performance.

5.5.5 Discussion

I generate the correlation over time graphs between the sensor readings and a reference sine wave to determine the length of time needed to adequately detect a user syncing. In general, the goal is to find the least amount of time needed to converge past a detection boundary. During the first evaluation, I demonstrated based on offline data that participants achieved separable correlation over time results. The goal of the second evaluation was to empirically demonstrate syncing accuracy with a live system and compare the time to trigger between touch swiping using the second hand.

Overall, the SynchroWatch technique is slightly slower than accessing the touchscreen with the second hand. During the second study, it took approximately 2.1 seconds for participants to complete the swipe action. Using a conservative correlation threshold of 0.9, it took participants approximately 5.5 seconds to sync while sitting and 6 seconds to sync while walking. However, by setting the coefficient to 0.8, offline analysis post-hoc indicates that a sync time of approximately 3 seconds is possible to achieve while sitting

and walking.

5.6 Demonstration Applications

I have developed several example applications that help illustrate the usefulness of SynchroWatch. The applications highlight the use of SynchroWatch for response and command gestures. See Figure 5.18.

Figure 5.18: Figures highlighting three demonstration applications: answering/dismissing a phone call, scrolling through a textview, and controlling a music player.

Dismiss Phone Call: Quick access to view and respond to notifications is one of the most compelling features of a smartwatch. After a notification prompts the user, a *response* gesture can be used to select between two actions. In the phone call application, two flashing icons to answer the call or send to voicemail are blinking out of phase on the screen. The user declines the call discretely by syncing with their thumb to the flashing decline icon, without even needing to lift the second hand. This demonstrates the use of SynchroWatch for binary selection.

Text Viewer: The second application uses SynchroWatch as a *command* gesture to scroll

through an article. I have two flashing targets at the top and bottom of the screen. Syncing to the bottom target scrolls the page down, while syncing to the top target scrolls up. It is possible to relax the threshold for scrolling to enable continuous scrolling after syncing. In other words, the user could extend and reposition their thumb until the application begins to scroll, then either maintain the thumb extended to stop or repositioned to continue scrolling. This facilitates a sync and hold interaction and does not require continuous syncing that may lead to fatigue.

Music Player: I increase the expressivity of SynchroWatch beyond binary target selection by introducing a wrist flick gesture. The wrist flick flips between two (or more) pairs of blinking controls. In the music player application, I present two pairs of blinking icons toggling between a vertical and horizontal layout, aligned similar to a D-Pad. The top and bottom targets are used to increase and decrease the volume, respectively. The left target is used to play/pause and the right target is used to transition to the next song. Using a separate gesture as a mode switch allows quickly toggling through a series of targets, which may be useful in other multi-target applications such as a messaging app (e.g., star, delete) or an application launcher.

5.7 Limitations and Future Work

There are a number of avenues I could explore to extend the capabilities of SynchroWatch. Allowing more than two targets is an obvious way to increase the expressivity of the technique. However, there is a balance between the number of moving targets visible on the small screen and the usability with the increasing distraction of additional targets. One approach is to have multiple targets blinking in parallel at unique frequencies. Multiple reference waveforms would have to be generated, complicating the detection algorithm and potentially increasing false positives. Another approach is to use a round-robin blink sequence. A round-robin approach, used in the current implementation, may prove less distracting to the user with only a single target blinking at a time. However, increasing the

number of targets will also decrease their observable frequency potentially slowing down the interaction. An advantage of the technique is that the reference sinusoidal waveform could be generated using any known frequency.

SynchroWatch currently supports syncing between two blinking targets that are out of phase at three frequencies (1.33 Hz, 1 Hz, 0.8 Hz). Some participants expressed difficulty syncing to a target blinking at 1.33 Hz, especially during the walking task. One idea is to adjust the frequency of the blinking controls depending on the activity context. For example, typical walking frequencies range from 1-2 Hz [10]. Based on motion sensor data from the smartwatch, the system could determine the ideal blinking frequency to increase the separation from the activity (e.g., walking). Alternatively, a simpler approach is to allow the user to specify the desired blinking frequency. In this case, muscle memory may enable eyes-free interaction when prompted.

Another way to enable eyes-free interaction is to leverage other feedback channels. For example, haptic feedback on the smartwatch or using a bone conduction actuator on a pair of smart glasses. Acoustic feedback can also be provided using a pair of earbuds. Visual feedback may also be extended by disaggregating the input and output feedback location (i.e., watch screen). Visual blinking controls could be displayed using the screen or a pair of LEDs in a head-up display, or on a nearby wall display in a smart environment. In this case, the smartwatch device is used only for sensing and timing between devices should be managed appropriately.

The recognition results reflect the system's performance in a semi-controlled environment with syncing during three tasks: relaxing while watching videos, actively browsing on the computer, and walking. Given the goal of achieving an interface that operates while the person is moving, the technique leverages magnetic sensing. However, a practical disadvantage is that some users may be disinclined to wear a passive magnetic ring. One advantage of the movement correlation technique is that it can be extended to operate with the accelerometer and gyroscope. I observed that the thumb extension and reposition ges-

ture induced slight movement in the accelerometer and gyroscope but not sufficient for detection, without using gross movements. However, an alternate gesture design (e.g., raising/lowering or rotating the wrist) matching a given blink frequency may be used without the need of a passive ring. Another caveat is that I intuitively suspect that using the accelerometer and gyroscope will also be less robust to global motion (e.g., walking), but I have not empirically validated the claim.

5.8 Contributions

I present a summary of the SynchroWatch system and its contributions using this dissertation's research goals. Figure 5.19 presents a summarized version.

Goal 1: Achieving *expressiveness* through a spectrum of interaction events and wearable interfaces

- Type of Interface and Purpose of Interaction

  SynchroWatch is a synchronous gesture interface. The system uses rhythmic mimicry of thumb movements to on-screen stimuli which can be detected within 3 seconds and used for notification-response and command gestures.

- Number of Unique Input Actions

  2 to 4 events are currently supported. The technique is designed to support binary selection of two blinking targets. A separate flick of the wrist gesture can be used to alternate between two pairs of targets for selection of up to 4 events.

- Information Transfer Rate

  The information transfer rate is 0.18 bps. The value is relatively lower than other systems based on the number of input actions supported (i.e., only binary selection).

- Applications

I've built a set of smartwatch applications for: answering and dismissing a phone call, scrolling through text or UI cards, and controlling your music player.

Goal 2: Enabling rapid *speed* of interaction and reducing the time between intention and action

- Time Between Intention and Action

  The time between intention and action (i.e., setup time) is 1.5 to 2 seconds. The current interaction requires visual stimulus. The user must raise the arm to field of view resulting in up to 2 seconds of setup time.

- Speed of Interaction

  The interaction can be completed within 3 seconds or more. A correlation-based technique with a threshold and user performance determines the time to trigger the binary selection. SynchroWatch is slightly slower than touch swiping but only requires one hand.

Goal 3: Expanding the *practicality* of one-handed input techniques and form factors

- Level of Instrumentation

  Level of instrumentation is medium. Rhythmic thumb movement can be captured using only a passive magnetic ring worn on the thumb, which alters the magnetic field around the smartwatch. No battery power or wireless connectivity is necessary.

- Prototype Readiness for Deployment

  The prototype readiness for deployment is rated TRL 5, which corresponds to a large scale prototype used in its intended environment. A 3D-printed case houses a neodymium magnet and could be designed as a fashionable accessory in the future.

Goal 4: Building *robust* detection approaches for multiple users and everyday activities

- Accuracy of Technique

  80-85% accuracy with correlation threshold within 3 seconds. See full details in the chapter.

- Usage Across Multiple Users

  SynchroWatch is a user-independent system. Detection of thumb synchronous correlation offline and online is dependent on correlation coefficient thresholds that are user-independent.

- Support During Everyday Activities

  High support. Detection of thumb synchronous correlation offline and online is robust during walking, browsing, and sitting activities. The detection is only supported when the arm is raised and interface is in the user's field of view.

5.9 Summary

SynchroWatch is an interaction technique for one-handed gestures on smartwatches that relies on *synchrony* between blinking on-screen controls and motions performed by the user. In particular, the system focuses on capturing continuous thumb extension and reposition movements matching on-screen flashes at a given frequency. The technique uses magnetic sensing with a passive magnetic ring on the thumb. The relative motion of the thumb ring induces a measurable vector change in the magnetic field around the smartwatch, which is correlated to the known stimuli frequency. I evaluated the performance of the technique using time-shifted correlation with a reference signal based on a physics model. The technique is calibration-less resulting in user- and device-independence. The technique is robust to noise and global motion enabling one-handed input while walking. I evaluated the technique initially offline during three tasks with varying degrees of hand and finger movement: walking, browsing on the computer, and sitting while watching a movie. I then evaluated the technique during a second study with an online version of the system, pro-

viding users feedback on their performance. I also compared the selection of binary targets with SynchroWatch versus the current mode of interaction using touch swiping. Results demonstrated that SynchroWatch is a viable input modality requiring only one hand, while touch swiping is faster to complete a selection but requires the use of the second hand. Finally, I concluded and demonstrated the opportunities enabled by SynchroWatch with various potential use cases. SynchroWatch addresses research goals focused on *expressiveness, speed, practicality* and *robustness*, and its contributions are summarized in Figure 5.19.

Dimensions	SynchroWatch
Goal 1: Achieving expressiveness through a spectrum of interaction events and wearable interfaces	
Purpose of Interaction	Command Notification Response
Number of unique input actions	Continuous thumb movement to enable binary selection
Information transfer rate	0.18 bps
Applications	
Goal 2: Enabling rapid speed of interaction and reducing the time between intention and action	
Time between intention and action	1.5 to 2 seconds
Speed of interaction	~3 seconds (threshold dependent)
Goal 3: Expanding the practicality of one-handed input techniques and form factors	
Level of instrumentation	Medium
Prototype readiness	TRL 5 Large scale prototype (intended environment)
Goal 4: Building robust detection approaches for multiple users and everyday activities	
Accuracy of technique	85% for syncing within 3 seconds
Usage across multiple users	User-independent
Support during everyday activities	High

Figure 5.19: Summary of contributions and research goals for SynchroWatch.

CHAPTER 6

WHOOSH: NON-VOICE ACOUSTICS FOR LOW-COST, HANDS-FREE, AND RAPID INPUT ON SMARTWATCHES

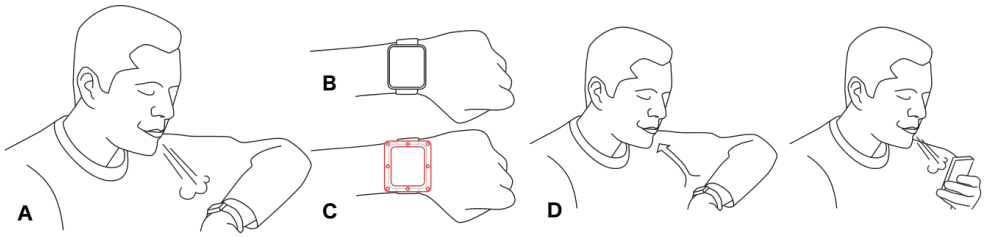


Figure 6.1: (A) Whoosh is an interaction technique that captures non-voice acoustic input (e.g., blowing, shooshing, other dynamic events), (B) using a commodity smartwatch without modifications and (C) with a custom-designed passive watch case. (D) The technique enables low-cost and rapid input, including multi-device events such as "sip" on the watch and "puff" on the phone.

6.1 Publication

This chapter is an extension of a full paper published and presented by Reyes et al. at the ACM International Symposium on Wearable Computers (ISWC 2016) [93].

6.2 Abstract

In this chapter, I present an alternate approach to smartwatch interactions using non-voice acoustic input captured by the device's microphone to complement touch and speech. *Whoosh* is an interaction technique that recognizes the type and length of acoustic events performed by the user to enable low-cost, hands-free, and rapid input on smartwatches (see Figure **??**). I built a recognition system capable of detecting non-voice events directed at and around the watch, including blows, sip-and-puff, and directional air swipes, without hardware modifications to the device. Further, inspired by the design of musical instruments, I developed a custom modification of the physical structure of the watch case to

passively alter the acoustic response of events around the bezel; this physical redesign expands the input vocabulary with no additional electronics. I evaluated the technique through two user studies in-the-laboratory and an initial study in-the-wild. The first study was conducted with 8 users — the unmodified watch condition with 10 events exhibits up to 90.6% accuracy. Using the watch case with 14 events, the system achieves 91.3% accuracy. These results are for user dependent machine learning models trained with a support vector machine (SVM), evaluated using ten-fold cross validation. I also present preliminary user independent results at 71.3% for the unmodified watch case and 79.7% for the instrumental watch case. A second user study with 10 participants is focused on evaluating and improving performance with user independent models, enabling the technique to potentially be used without user training a priori. I compare results using two techniques — the support vector machine (also used in the first study) and a Hidden Markov Model (HMM) pipeline using HTK/GT2K. Overall for the unmodified watch and the watch case respectively, the system achieves 82.1% and 84.3% for the SVM pipeline and 79.9% and 78.6% for the HMM pipeline. I also evaluated the performance of the system in-the-wild with 4 participants. Finally, I share a number of demonstration applications, including multi-device interactions, to highlight the technique with a recognizer running on the watch.

6.3 Introduction

The emergence of smart devices (e.g., mobile phones, smartwatches, and head-up displays) is redefining the way we access data and produce information through everyday microinteractions [6], interactions that take less than four seconds to initiate and complete. The primary input modality for the smartwatch and mobile phone is touch. Touch offers expressive multitouch capabilities and is intuitive. For example, recent work demonstrates the possibility of performing text entry with a smartwatch on-screen keyboard, using statistical decoding and error correction [32]. However, touch input on the small screen of a watch still requires targeted visual attention and a free hand for interaction. Traditionally,

occlusion and fat-finger selection errors are two common challenges that hinder the use of these small screens [56, 101].

With advances in connectivity and computing, phones and smartwatches are capable of near real-time speech recognition. Speech provides a fluid and hands-free way of communicating intent and commands to a smart device. However, speech may be tedious and not well suited to certain microinteractions, such as repetitive input, scrolling, or swiping. In this chapter, I present an approach to smartwatch input using non-voice acoustics to supplement touch and speech. The input modality opens up opportunities for hands-free input on small screen devices and also has implications for assistive technologies. The "sip-and-puff" technique is popular for wheelchair controls [27] and inspires some of this work. The event set includes blow events as well as other acoustically unique sounds (e.g., shoosh, double blow, long blow) produced by modulating the shape of the mouth, tongue and throat. Non-voice acoustic input on the smartwatch can be subtle, and with the device in proximity to the mouth, can also be performed quietly or in environments with high ambient noise.

The event set and applications are designed around familiar metaphors from mouse, touch and physical interactions. This design consideration facilitates mapping non-voice acoustic events to intuitive actions on the device. For example, a localized blow on the bezel can be used to click a corner icon, air swipes are useful for directional commands in the interface, and sip-and-puff is used to "absorb" content and "deliver" it to another device or application. The system runs in real-time and can be installed on commodity mobile platforms that are equipped with a microphone. Further, through a completely passive watch case modification, our design can facilitate robust recognition of an expanded set of input events.

This chapter makes the following contributions:

- I describe an interaction technique using non-voice acoustic input for smartwatch interactions that enables low-cost, hands-free, and rapid input.

- I introduce the use of passive, 3D-printed smartwatch cases to expand the expressivity of events by introducing air swipes, circular blows, and bezel blows.
- I provide empirical evidence of the recognition system performance and limitations, through an initial lab study with 8 participants, an evaluation with 4 participants in-the-wild, and a follow-up evaluation with 10 participants in-the-laboratory.

6.4 Interaction Techniques with Unmodified Watch

I describe the space of input events I explore with an unmodified smartwatch equipped with a single monophonic microphone, and draw analogies to common mouse and touch inputs.

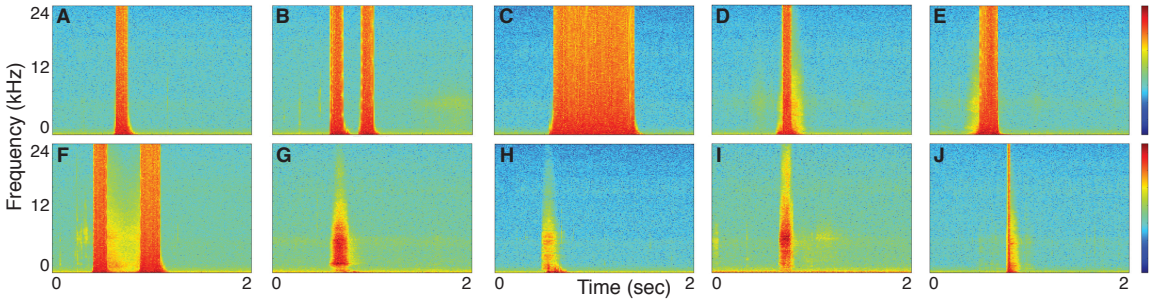


Figure 6.2: Spectrogram figures for unmodified watch events. The events displayed are: (A) short blow, (B) double blow, (C) long blow, (D) swipe up, (E) swipe down, (F) clockwise blow, (G) shoosh, (H) open exhale, (I)-(J) sip-and-puff.

6.4.1 Directed Blows

The *short blow* is the most basic event in the set with a quick blow toward the center of the watch screen. Based on empirical data from pilot studies, the typical length of a short blow is 200ms. In general, the spectrogram for this event (shown in Figure 6.2A) exhibits saturation when blowing normally with the device close to the mouth. During evaluation, I observed 10-20cm to be the typical distance between the mouth and the device while blowing at the smartwatch. The *double blow* extends the recognition of a discrete blow while capturing consecutive short blows directed at the center of the screen, lasting between 400-500ms (see Figure 6.2B). The double blow is analogous to a double click. The *long*

blow consists of a continuous blow aimed at the center of the watch screen, typically longer than 500ms based on pilot study data (see Figure 6.2C). The long blow is analogous to a press-and-hold interaction from touch events.

6.4.2 Air Swipes

Similar to swipe gestures in touch-based interactions, air swipes are directional events captured as air passes over the watch screen and wind noise is captured by the microphone (located at the bottom center of the bezel). Typical length for swipes is 300ms. A *swipe up* is a continuous blow from bottom-to-top of the bezel across the screen. Conversely, the *swipe down* begins at the top of the bezel and ends at the bottom. See Figures 6.2D-E. A *circular blow* is a continuous swipe performed in a clockwise direction around the bezel. The blow starts and ends at the bottom center of the bezel, where the microphone is located. Based on training data, I observed circular blows lasting up to 1 second (see Figure 6.2F).

6.4.3 Non-Voice Sounds

A *shoosh* sound is produced by modulating a blow with curled tongue and pursed lips. A shoosh is typically used to indicate a form of silence or dismissal in the interface. Typical length is about 200ms (see Figure 6.2G). An *open-mouth* consists of the user exhaling toward the watch screen with their mouth open. This action is similar to fogging your eyeglasses with your breath. Typical length is about 300ms (see Figure 6.2H). A *sip* is performed with pursed lips similar to using a drinking straw. Compared to other events, the sip is an inhale and can be used to indicate directionality away from the device. A *puff* accompanies the sip event. A strong "p" sound distinguishes the puff from a short blow. Typical length for sip-and-puff is about 200ms (see Figures 6.2I-J).

6.5 Whoosh

Whoosh runs in real-time on the smartwatch and performs audio recognition on incoming microphone data.

6.5.1 Theory of Operation

The main parts of voice and acoustic production are the lungs, the larynx or vibrator, and the resonator system. Air is exhaled out of the lungs and passes through the larynx, which contains the vocal folds. For blow events, air passes through relaxed folds and lung capacity determines the forcefulness of the blow. For non-voice sounds, the airstream passes between the vocal folds as they vibrate between 100Hz to 1kHz. The muscles in the larynx control the pitch based on the length and tension of the vocal cord. As the folds vibrate, they produce a buzzing sound at different frequencies, similar to the mouthpiece of a trumpet. The resonator system, consisting of the throat, nose, and mouth, alter the pathway to produce human speech and other sounds, similar to the structures of a musical instrument.

The system focuses on both the wind noise detected by the microphone while blowing, as well as non-speech human sounds. Depending on the forcefulness of a blow or non-voice event, proximity to the microphone, and the direction of the user's mouth, different phenomena is exhibited. A *blow* event may produce either a broadband frequency response through the microphone or exhibit distortion from clipping caused by non-linear behavior of the electronic components and power supply limitations (Figures 6.2A-F). The system uses this distortion to its advantage to minimize false positives and uniquely identify particular events. Other events such as *shoosh*, *open*, and *sip-and-puff* exhibit distinct spectral patterns with energy up to approximately 10kHz (Figures 6.2G-J). After isolating an event and extracting features based on its frequency response, the system infers the type and location of the event based on pre-trained audio fingerprinting using a machine learning classifier.

6.5.2 Implementation

I use an LG G (Android) Watch with a single microphone located at the bottom center of the bezel of its 1.65 inch touch screen, as well as a Motorola Droid Turbo (Android) smart phone to explore multi-device interaction. The microphones on both devices are sampled at 48kHz using the default microphone source, 16-bit PCM encoding, with no audio gain or noise suppression. TarsosDSP[1] handles audio management and recording. The library delivers a *float[]* audio buffer at preset frame intervals for processing in real-time.

Segmentation: This task focuses on determining when an event of interest occurs within the audio stream. For training purposes, users are prompted to provide input and audio is recorded for each event individually in 2-second windows. In the offline analysis, I implement a form of silence detector using a rolling variance to isolate the beginning and end of the event in each audio file. The pipeline uses a forward and backward threshold search to account for events with silence occurring during the event (e.g., double blow), and empirically determine a threshold robust to noise around the event. I then expand the trimmed window outward around the isolated data by a preset number of frames to ensure full capture of the event and pass it to the feature extractor. For the real-time pipeline, I use a silence detector based on the energy of an audio frame (approximately 20ms). I maintain a buffer of audio frames that comprise an input event and use a heuristic timing threshold when silence is detected during an actual event (e.g., double blow). Once the event input buffer is full, the audio data is passed to the feature extractor.

Feature extraction: In order to capture directional events, I divide the segmented signal into two window slices of equal length. Dividing the audio signal in two halves aids in capturing salient features occurring about the center of the event. Mel-frequency cepstral coefficients (MFCCs) are a set of acoustic features modeling the human auditory system's non-linear response to different spectral bands. I calculate a 26-dimension MFCC with band edges from 0Hz to half the sampling rate at 24kHz. I calculate the sum of each MFCC

[1] https://github.com/JorenSix/TarsosDSP

coefficient for all frames (20ms frame, 10ms overlap) in each half of the audio signal, with the energy as the first coefficient. The MFCC vectors for each half add up to a total of 52 features. I use an additional 26 features based on the deltas of the MFCC coefficients. The features are normalized for classification. I run principal component analysis (PCA) on these features to facilitate the real-time classification.

Classification: I use a support vector machine (SVM) algorithm trained using Weka's sequential minimal optimization (SMO) implementation with a cubic polykernel and default parameters.

6.6 Discussion & Limitations for Unmodified Watch

During the initial exploration into the design of the event set, I experimented with swipes in all directions (i.e., up, down, left, and right). Given that I use a smartwatch with a single front-facing microphone at the bottom center of the bezel, I found intuitively that the location of the microphone was key to discriminating between events. A continuous blow approaching from the left or the right and passing over the microphone appear symmetrical, and thus are difficult to discriminate in the recognizer. In contrast, swipes up and down begin either at or away from the microphone, making it easier to recognize them as unique events.

Additionally, I also experimented with localizing directional blows on the arm to the left and right sides of the watch, as well at the top, bottom, left, and right target areas on the bezel. In prior work [86], researchers localize up to 5x5 events with a single microphone on a laptop screen roughly an order of magnitude larger than a smartwatch screen. However, on our platform, the single microphone and small size of the watch did not provide the ability to readily disambiguate such inputs. To address these limitations and expand the Whoosh vocabulary, I design a custom 3D-printed watch case.

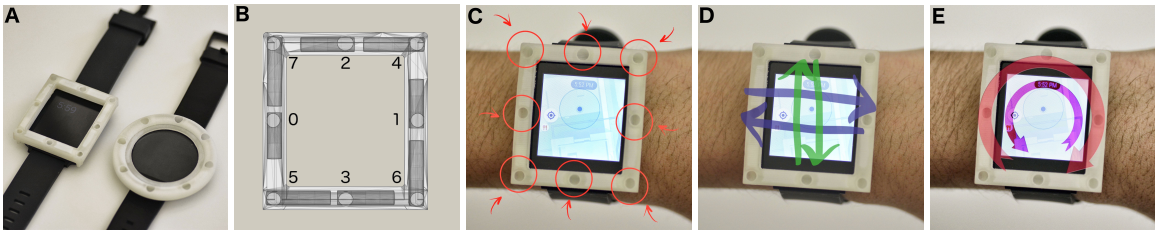


Figure 6.3: FluteCase and interactions. (A) FluteCase designs for both square and round watch faces, (B) Translucent rendering of the square FluteCase's 3D model with tube labels, (C) Bezel blows, (D) Air swipes, and (E) Circular blows.

6.7 FluteCase: A Passive 3D-Printed Watch Case

FluteCase is a custom 3D-printed watch case for both square and circular smartwatches that alters the acoustic response of blowing events on and around the smartwatch. The case provides a low-cost and entirely passive (meaning no electronics nor battery usage) means of expanding the range of inputs that are recognized by the system. In this section, I describe the phenomena and inspiration for the design of a passive modification of the physical structure of the watch, based on wind instruments and prior work altering the speaker-microphone pathway on mobile phones [62].

6.7.1 Acoustic Phenomena

When air is blown into a tube-shaped resonator, standing waves are created that cause the air to vibrate and produce sound. For closed pipe wind instruments like ours, the pitch of the vibration is determined by the length of the tube. For example, the Greek pan flute has multiple tubes with different lengths open at one end for blowing and is closed at the other end. Closed pipe resonators do not require finger operation and their fundamental air resonant frequencies are defined by:

$$f = \frac{v}{\lambda} \qquad \text{[Hz]} \quad (6.1)$$

where f is the air resonant frequency, v is the speed of sound, $\lambda = kL$ is the wavelength, where k is a constant determined by open or closed pipe and L is the length of the pipe. Generally, the shorter the pipe is, the higher the resonant frequencies produced.

6.7.2 Design of the FluteCase

I draw inspiration from the structure of closed pipe instruments to design the 3D-printed FluteCase. I develop both square and round watch versions to suit a variety of commercial devices (see Figure 6.3A). The cases have 8 closed pipe tubes of different lengths, each with an open hole. The tubes' "head" (the end with the open hole) and "tail" (the closed end) are connected to each other. In the case of a circular smartwatch, the head and tail form a ring shape around the watch display. A base that fits the shape and size of the watch bezel attaches tightly to the watch. The eight tubes are designed to resonate at eight distinct frequencies between 2kHz to 10kHz, allowing blows near particular regions of the watch face to be readily disambiguated. For replicability, I describe the dimensions of the square case used during the user evaluation. The overall width, length, and depth of the square case are 45.60 mm, 51.06 mm, and 5.58 mm, respectively. The diameter of each hole is 4.05 mm. The width of each circular tube is constant at 4.096 mm. The length of each tube is defined by:

$$L = 14.956 * 2^{\frac{i}{12}} \quad \text{[mm]} \quad (6.2)$$

where L is the length of each tube as a function of i, which denotes the ith tube (labeled in Figure 6.3B).

6.8 Interaction Techniques with FluteCase

The FluteCase design greatly expands the range of non-voice acoustic interactions with smartwatches, allowing recognition of an additional 6 swiping blows and 8 bezel blows. A blow event over each FluteCase hole creates a slightly audible tone generated by the

airflow entering the resonator tube. I use the same recognition pipeline described for the unmodified watch scenario, as the segmentation is adaptive to variable event lengths. Bezel blows and swiping blows are shown visually in Figures 6.3C-E.

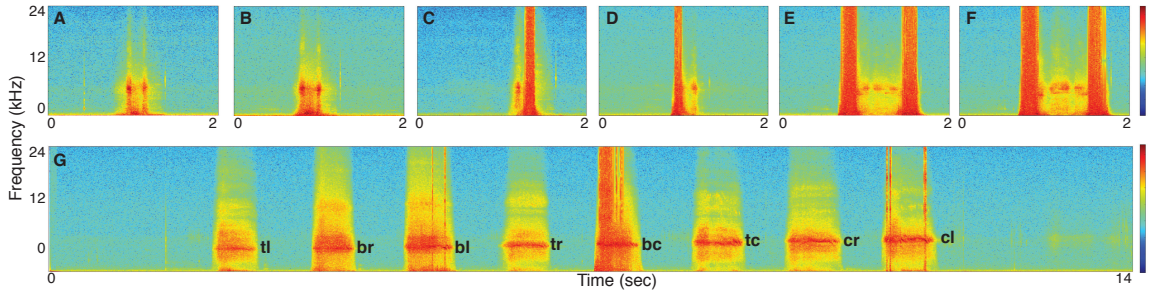


Figure 6.4: Spectrograms for FluteCase events: (A) swipe left, (B) swipe right, (C) swipe down, (D) swipe up, (E) clockwise, (F) counterclockwise, (G) bezel blows (labeled on the figure) starting from the lowest to highest resonant frequency.

6.8.1 Swiping Blows

Blowing over two or more FluteCase holes in a swiping fashion enables six additional input events. *Air swipes* are single blows across the watch face traversing two holes in the following directions: left-right, right-left, top-bottom, or bottom-top. *Circular blows* are swipes along the edge of the watch, traversing all holes, in a clockwise or counterclockwise direction beginning at the bottom center location. The spectrograms for all swiping blows are shown in Figures 6.4A-F.

6.8.2 Bezel Blows

Bezel blows are discrete events performed by the user in a single-action blowing at one of the eight holes distributed evenly around the watch case. *Corner* bezel blows consist of a continuous blow at one of the four corner targets of the watch case. These events are: topleft (tl), topright (tr), bottomleft (bl), bottomright (br). The next set of events are *D-pad* bezel blows. In this spatial arrangement, a continuous blow is directed at FluteCase target locations emulating a D-pad keypad configuration. These events are the remaining

bezel locations: topcenter (tc), centerleft (cl), centerright (cr), and bottomcenter (bc). The spectrogram for bezel blows starting from the lowest to highest resonant frequency is shown in Figure 6.4G.

6.9 Evaluation I - Initial Laboratory Study

6.9.1 Study Design

I conducted a technical evaluation of the interaction techniques with an unmodified watch in the usability lab of our institution. Eight participants (5 male, 3 female, ages 22-34) were part of the user study. Participants wore the watch on the left hand. To begin, researchers performed a demonstration of each technique. Participants familiarized themselves with the data collection application with a practice round. During the evaluation, a visual stimulus was presented to the participant on the watch screen prompting them to perform a given event. Audio was recorded for 2.5 seconds after the prompt, with a one second pause between events. Each event was recorded in an individually labeled audio file for segmentation. The participants performed four rounds of data collection. Participants were allowed to rest, drink water, remove the watch, or leave the room between sessions if desired.

For the unmodified watch, I evaluated the system for 10 events. In each round, participants performed 5 examples for each event. The order of the stimulus was randomized across each round. In summary, the user study included 8 participants x 10 events x 4 rounds x 5 samples per event for a total of 1600 event samples. I discarded a total of four instances where the researcher observed the participant perform the wrong event or the segmentation determined the event was not fully captured within the time window.

I conducted a technical evaluation of the new event set with a FluteCase-mounted smartwatch using the same device, participants, and pipeline as the previous section. The data collection for this condition included 8 participants x 14 events x 4 rounds x 5 samples per event for a total of 2240 event samples. I discarded a total of 61 instances (roughly less than 5 out of 160 samples per event) in which participants either perform the wrong event or the

event is not fully captured within the time window. I evaluated how the system performed on a per-user level using 10-fold cross validation and how it generalizes across participants using leave-one-participant-out analysis.

6.9.2 Per-User Classifiers

I evaluated the technique by applying 10-fold cross validation on each individuals' collected instances with a SVM polykernel. For the unmodified watch condition, the overall average per-user accuracy across 8 participants and 10 events is 90.6% (sd=7.3%). P1 achieved the highest accuracy at 98.5% and P5 achieves the lowest accuracy at 85.5%. I observed that P5 held the device farther from the mouth than other participants, roughly more than 20cm. The distance away from the mouth results in weaker signals at the microphone making it difficult to distinguish between events. The confusion matrix of the results is shown in Figure 6.5. The lowest precision of 78.1% was observed for the double blow event, mostly confused with a short blow. In some cases, participants performed the double blow quickly, effectively being recognized as a short blow. The shoosh event achieved the highest accuracy at 98.8%.

For the FluteCase condition, the average accuracy using 10-fold cross validation across 8 users and 14 events is 91.3% (sd=7%). P8 achieved the highest accuracy at 96.4% and P5 achieves the lowest accuracy at 80.4%. The confusion matrix of the results is shown in Figure 6.7. The lowest precision of 81.6% was observed for both the clockwise and counterclockwise events, mostly confused with each other. Both of these are compound gestures that require blowing over all eight bezel locations. The bottom center event is the most accurate with an accuracy of 99.4%. I suggest the main reason for the highest accuracy is that the microphone is located directly underneath the bottom center bezel hole, resulting in a clearer signal. Swipe down and swipe up event also presented accuracies above 95% suggesting they could be effective to control up/down actions in the interface.

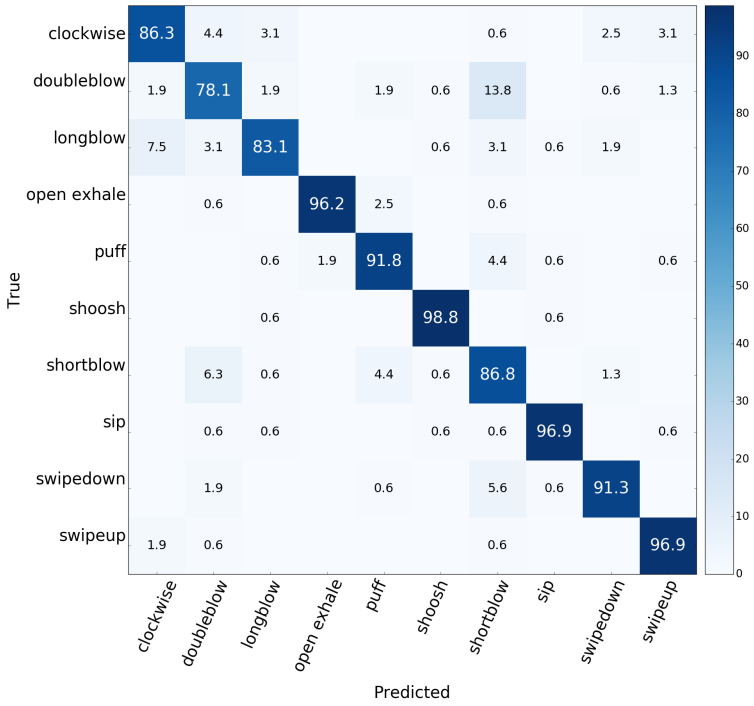


Figure 6.5: User dependent confusion matrix averaged across all users (in %) for the unmodified watch. Rows represent ground truth and columns are predicted values.

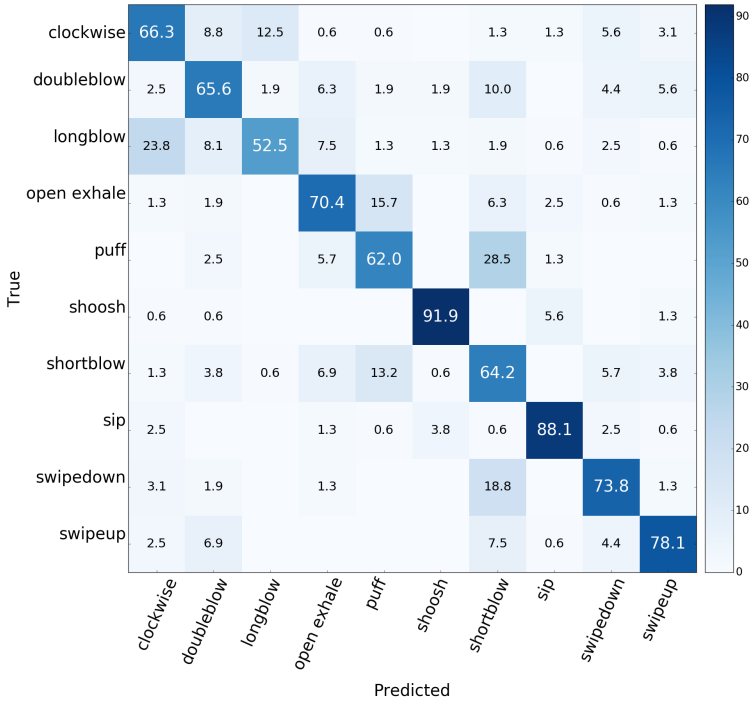


Figure 6.6: Leave-one-out confusion matrix averaged across all users (in %) for the unmodified watch. Rows represent ground truth and columns are predicted values.

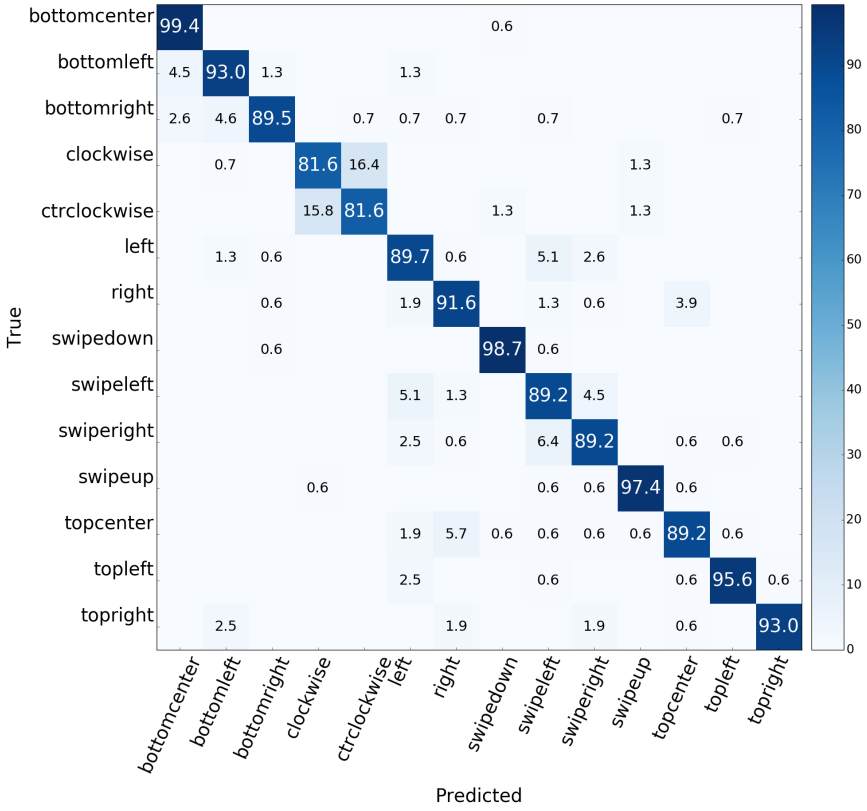


Figure 6.7: User dependent confusion matrix averaged across all users (in %) for FluteCase. Rows represent ground truth and columns are predicted values.

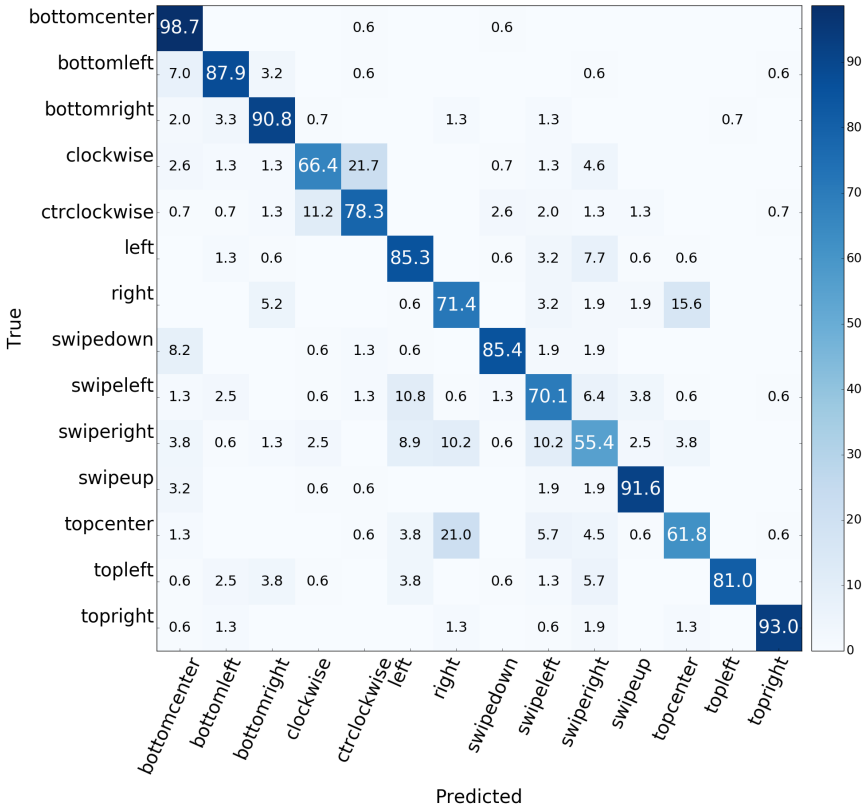


Figure 6.8: Leave-one-out confusion matrix averaged across all users (in %) for FluteCase. Rows represent ground truth and columns are predicted values.

6.9.3 General Classifiers

I also evaluated how the system generalizes across users with initial user independent models. Preliminary leave-one-participant-out analysis (i.e., test with one participant, train with the rest) is used with the SVM pipeline (linear polykernel).

For the unmodified watch condition, the system reported an overall accuracy of 71.3% (sd=7.2%) across 8 participants and 10 events. P7 achieved the highest accuracy at 80.7% and P4 achieved the lowest accuracy at 62.5%. The lowest precision of 52.5% was observed for the long blow event, mostly confused with a clockwise event likely based on the similar time lengths between the events. The shoosh event achieved the highest accuracy at 91.9% with distinct high frequency energy as compared to other events. Sip achieved the second highest accuracy at 88.1%. Puff was highly confused with a short blow. For future event sets, I recommend using a short blow to replace a puff event. Anecdotally, participants were unsure about how to perform the event, plus its short length and saturation of the microphone leads to confusion with the short blow. The confusion matrix of the results is shown in Figure 6.6.

Similarly for FluteCase, preliminary leave-one-participant-out analysis across 8 participants and 14 events results in overall accuracy of 79.7% (sd=9.9%). P4 achieved the highest accuracy at 88.2% and P5 achieved the lowest accuracy at 59.2%. The lowest precision of 55.4% was observed for the swipe right event, mostly confused with a swipe left and left events. In some cases, participants may not blow directly into or over the intended FluteCase hole, leading to confusion. The bottomcenter event achieved the highest accuracy at 98.7% with the microphone located at the bottom bezel. There was also significant confusion between the clockwise and counterclockwise events. Completing these events requires blowing over all FluteCase holes in either direction. The classifier expects to see a sequence of increasing or decreasing resonant frequencies which are not present if the user misses one of the holes. The confusion matrix of the results is shown in Figure 6.8.

6.10 Evaluation II - Activation Event In-The-Wild for the Unmodified Watch

I have demonstrated the recognizer is capable of discriminating events with the in-lab study and the technique is feasible on smartwatches available today with and without a modified watch case. This section focused on determining the feasibility of using a double blow event and an unmodified watch for activation events in-the-wild.

To minimize unintentional activation during real-world use, activation events are designed to distinguish intentional interactions from everyday activities. Activation events should ideally be extremely resistant to false positives while achieving high recognition rates [95]. Once the system is activated, all other input events are recognized. I design an activation trigger consisting of a double blow. While double blow presents some confusion with the classifier (see Figure 6.5), I believe it would be robust against detecting noise based on its acoustic signature. I present results for false negatives with double blow performed in-the-wild by 4 participants, and false positives with noise only recorded by 4 researchers.

6.10.1 False Positives (fp)

I recorded smartwatch ambient data where there is no intentional input from 4 researchers on this project. In total, I recorded 11hr:36min of ambient audio at 48kHz sampling rate. The dataset was meant to be representative of a "day in the life" of a smartwatch wearer. The audio recordings captured audio walking outdoors, meetings occurring indoors, and other ambient noise. The data collection was limited only by the battery life of the smartwatches used, and the longest recording was approximately 3 hours on a LG G Watch. I apply a highpass filter above 15kHz to isolate any activation event that exhibits clipping and remove most ambient noise below 4-7kHz. I then apply a peak detection algorithm to identify double blow events. The recognizer mislabels noise as a double blow event 15 times, resulting in a 1.3 fp/hour rate. Most confusion was observed during hand washing

at the sink and several forceful coughs near the device. Figure 6.9 highlighs false positives detected during hand washing at the sink and several forceful coughs near the device. It is possible to mitigate false positives by only listening for command gestures for a short and fixed period of time after activation. The system can then quickly deactivate to decrease the chances that a false positive command is detected after a false positive activation.

6.10.2 False Negatives (fn)

I recruited 4 participants from the first technical evaluation to perform the double blow activation gesture in-the-wild. I asked each participant to wear the watch for at least 4 hours during the day and perform the double blow when prompted. An application on the device prompted participants by vibrating the watch and presenting on-screen instructions to perform the blowing event. Prompts were delivered using a random Poisson process with an average delivery time of 4 minutes. To preserve battery and privacy, the application recorded only one minute of audio data after prompting the user. In total, the four participants were prompted 174 times. I discarded 22 missed instances where the participant did not perform the event, based on visual analysis and acoustic inspection of the data. The peak detection search algorithm correctly identified the double blow 149 out of the remaining 152 instances, resulting in 98.0% accuracy. Figure 6.10 highlights audio recording for a double blow event in-the-wild. The double blow saturates the microphone in the first few seconds of the prompt. Speech between a participant and another person are captured in the remaining one minute of audio.

6.11 Evaluation III - Laboratory Study

I conducted a second in-the-laboratory study to assess the repeatability of the results from the first technical evaluation. Additionally, this evaluation focused on enhancing the performance of the recognition system, primarily on enabling user independent or general classification. To that end, I introduced additional instructions to the participant with a

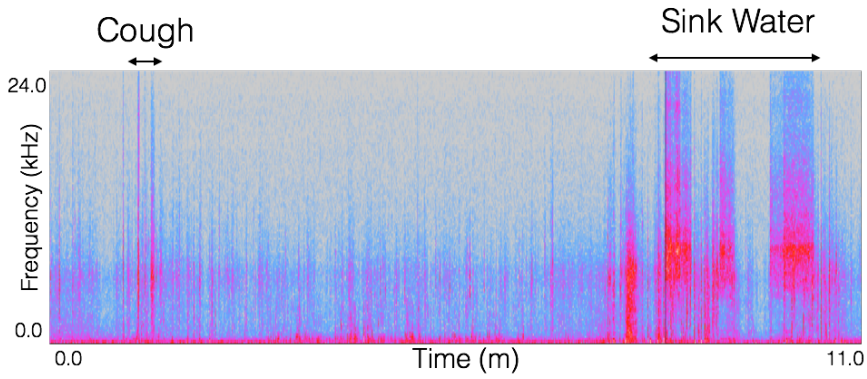


Figure 6.9: Figure highlighting false positives detected during hand washing at the sink and several forceful coughs near the device.

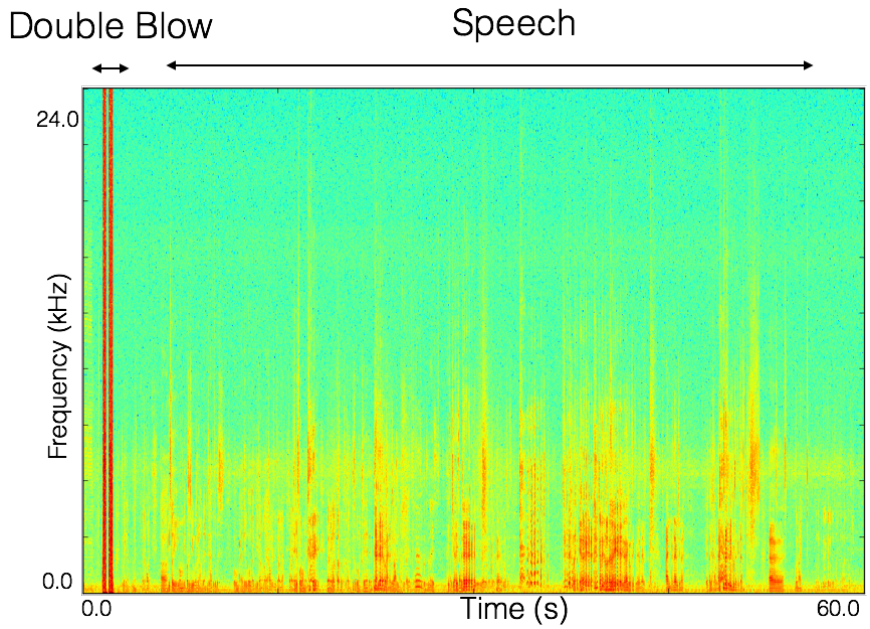


Figure 6.10: Figure highlighting audio recorded for a double blow performed in-the-wild. The double blow saturates the microphone as seen a few seconds after the prompt. The remaining audio file consists of speech data between the participant and a friend for one minute.

training video meant to be provide consistency and repeatability in performing events. I also compare the results of the SVM algorithm with a HMM-based modeling pipeline.

6.11.1 Study Design

I conducted a technical evaluation of the interaction techniques in the usability lab of our institution. Ten participants (7 female, 3 male, ages 18-25) were part of the user study. Participants wore a commodity LG G watch on the left hand. To begin, the researcher played an instructional video on a laptop in front of the participant which described proper arm/hand placement and demonstrated examples of how to perform each acoustic event. Participants familiarized themselves with the data collection application with a practice round. A researcher provided feedback as they perform each event. During the evaluation, a visual stimulus was presented to the participant on the watch screen prompting them to perform a given event. Audio was recorded for 3.5 seconds after the prompt, with a one second pause between events. Each event was recorded in an individually labeled audio file for segmentation. The participants performed four rounds of data collection. Participants were allowed to rest, drink water, remove the watch, or leave the room between sessions if desired.

For the unmodified watch, I evaluated the system for 10 events. In each round, participants performed 5 examples for each event. The order of the stimulus was randomized across each round. In summary, the user study included 10 participants x 10 events x 4 rounds x 5 samples per event for a total of 2000 event samples. I discarded a total of 138 instances where the researcher observed the participant perform the wrong event or the segmentation determined the event is not fully captured within the time window.

I conducted a technical evaluation of the event set with a FluteCase-mounted smartwatch using the same device, participants, and pipeline as the previous section. The data collection for this condition included 10 participants x 14 events x 4 rounds x 5 samples per event for a total of 2800 event samples. I discarded a total of 147 instances in which

participants either performed the wrong event or the event was not fully captured within the time window because the participant performed the event early or late.

6.11.2 Approaches to Enhancing Performance

Instructional Training Video

To ensure consistency of the instructions given across participants, I prepared an instructional video on how to acoustically and physically perform the acoustic events for both the unmodified watch and FluteCase. The instructional training video provides both visual and audio feedback on each of the acoustic events, and includes suggestions on the recommended proper placement of the arm (approximately 10-20 cm from the mouth). The video includes a researcher demonstrating how to perform each of the acoustic events three times from two different vantage points - side view and user-facing third person view. The video runs approximately 5 minutes long and was played for participants at the beginning of the evaluation. Figure 6.11A-C show video frames extracted from the participant training video.

Figure 6.11: Video frames extracted from the Whoosh participant training video. A researcher demonstrates how to use Whoosh. A) Frontal view to perform an acoustic event for the unmodified watch; B) Frontal view to perform an acoustic event for the FluteCase; and C) Side view shows the suggested placement of the arm 10-20 cm away from the mouth and how to perform a swipe down event.

Modifications to Data Collection Application

There are two modifications made to the smartwatch data collection application used in the previous laboratory evaluation. The modifications were designed to enhance performance and minimize errors by participants. The first modification is the expansion of the audio recording window from 2.5 seconds to 3.5 seconds. The change was made to ensure that the entire acoustic event could be captured within the time window. The application provides instructions on what acoustic event to perform and feedback on how many events have been completed. The second modification to the application is the addition of a progress bar indicating the elapsed recording time of 3.5 seconds. The user is instructed by the researcher to attempt to perform the acoustic event when the progress bar is approaching the center of the screen (or 50% of the recording window) to avoid audio captured too early or too late within the window.

6.11.3 Per-User Classifiers

I evaluate how the system performs on a per-user level using the previously described SVM pipeline. Features used in this section are MFCC coefficients with 20 ms window length and 10 ms overlap. I evaluate the technique by applying 10-fold cross validation on each individuals' collected instances.

For the unmodified watch condition using SVM (linear polynomial kernel), the best overall average per-user accuracy across 10 participants and 10 events is 91.8% (sd=7.3%). P9 achieved the highest accuracy at 96.9% and P1 achieved the lowest accuracy at 84.7%. I present the confusion matrix of the results in Figure 6.12. The shoosh event once again achieved the highest accuracy at 100%. The lowest precision of 83.2% was observed for the puff event, always confused with a shortblow. The two events are similar in length and appear the same if the microphone is saturated. There was some confusion between single and double blow. In practice, I observed the classification can be improved if there is a clear pause between each of the blows in the double blow.

For the FluteCase condition using SVM (cubic polynomial kernel), the best average accuracy using 10-fold cross validation across 10 users and 14 events is 92.7% (sd=7.2%). P5 achieved the highest accuracy at 97.5% and P7 achieved the lowest accuracy at 83.9%. I present the confusion matrix of the results in Figure 6.13. The lowest precision of 87.6% was observed for the bottom right which is mostly confused with bottom left due to its proximity to each other. Clockwise and counterclockwise exhibited confusion with each other. To perform these events, the user needs to ensure they blow into each of the holes as they traverse the bezel. Targeted blows and a slower movement around the bezel could help ensure higher accuracy. The bottom center event was the most accurate with accuracy of 100%. I suggest the main reason for the highest accuracy is that the microphone is located directly underneath the bottom center bezel hole, resulting in a clearer signal. Overall, all other events are accurate above 90%.

6.11.4 HMM Modeling

In this section, I present details on modeling the acoustic events using HMMs offline for user independent leave-one-participant-out analysis. Each audio file is trimmed using the variance detector described earlier in the chapter. Before passing the data to the HMM pipeline, I calculate features for all audio recordings collected during the second evaluation. A feature set of 26-dimension Mel-frequency cepstral coefficients (MFCCs) with band edges from 0Hz to half the sampling rate at 24kHz is calculated for every audio frame in the dataset. A standard window size of 20 ms and 10 ms overlap is used. A feature vector per file is calculated for the HMM pipeline. The rows indicate the number of audio frames in the recording after trimming for silence. The columns are the 26 MFCC feature coefficients calculated in each audio frame per row.

The feature vector, $\mathbf{x}$,

$$\mathbf{x} = (\mathbf{MFCC_1}, \mathbf{MFCC_2}, ..., \mathbf{MFCC_{26}}) \tag{6.3}$$

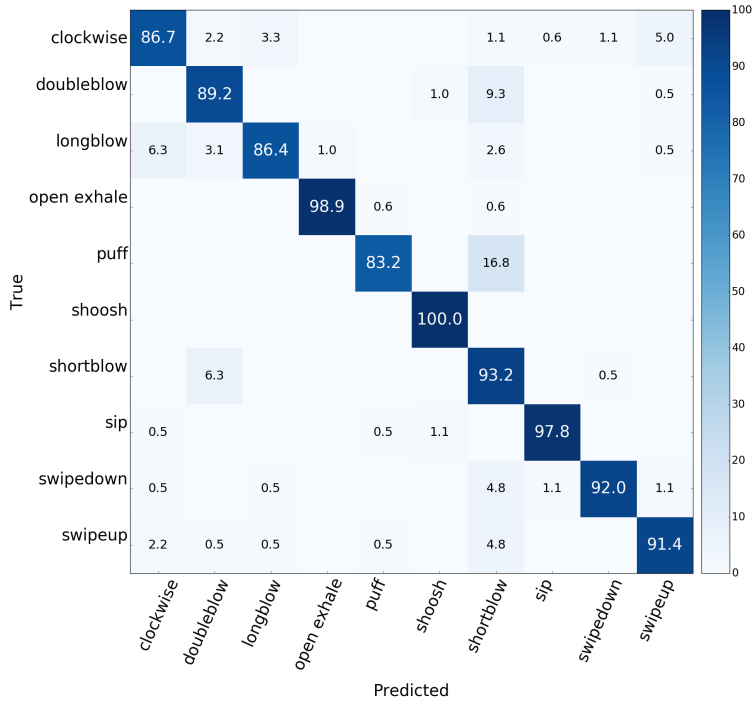


Figure 6.12: User dependent confusion matrix averaged across all users (in %) for the unmodified watch. Rows represent ground truth and columns are predicted values.

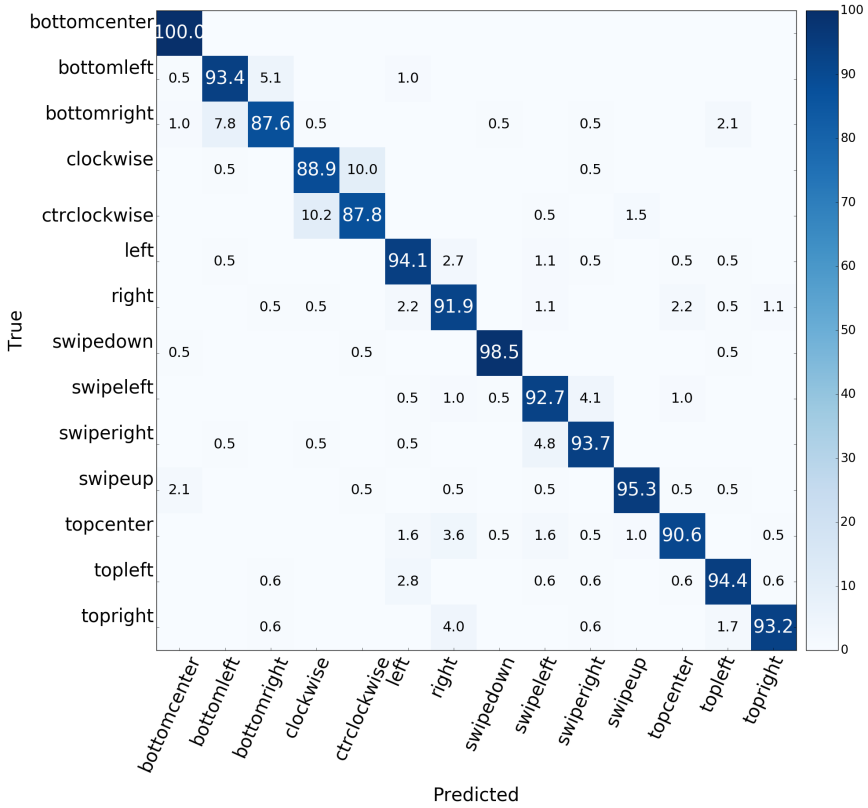


Figure 6.13: User dependent confusion matrix averaged across all users (in %) for Flute-Case. Rows represent ground truth and columns are predicted values.

is used for training the HMMs for the entire system. Here $MFCC$ is one of the cepstral coefficients calculated per audio frame. Training and testing of the HMMs is done using HTK 3.4.1 [120].

I use a single topology to model all acoustic events for both the unmodified watch and FluteCase. The topology consists of an 8-state HMM with skip states between states 1->5 and 5->8. A graphical representation of the HMM topology is shown in Figure 6.14. Although regular speech recognition utilizes three states per phoneme, the topology is designed to capture the complexity span between simple events such as a short blow with the unmodified watch case and more complex events such as a clockwise event with the FluteCase. For example, in the FluteCase condition, an acoustic event may require blowing sequentially over a series of bezel targets.

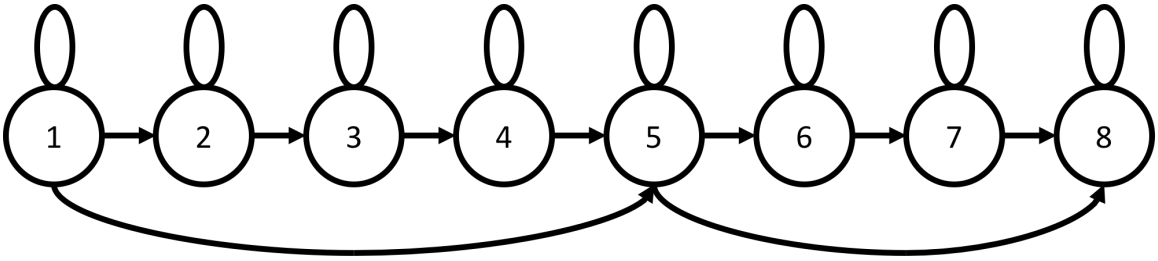


Figure 6.14: The HMM topology used for each acoustic event with 8 states and two skips between states 1->5 and 5->8.

To model each state, I used a Mixture of Gaussians (MoG). In a MoG HMM, the probability distribution function (PDF) learnt to represent each state is a weighted sum of Gaussians. This technique allows more freedom in approximating PDFs. I experimented with the number of Gaussians and ultimately used a single mixture to represent the data with a mean of 0 and standard deviation of 1. Due to a limited amount of training data, higher number of mixtures would not converge and could not be used. All states were represented using diagonal covariance matrices. The best set of parameters were chosen to finalize the models. The grammar used is shown in Equation 6.4. There are additional "START" and

"END" states to model the pause before and after the acoustic event is uttered.

$$grammar = (START\ \$phrase\ END) \tag{6.4}$$

6.11.5 General Classifiers

I also provide a comparison of how the system generalizes across participants using leave-one-participant-out analysis with two algorithms, the previously described SVM pipeline and a newly introduced Hidden Markov Model (HMM) pipeline described above. Below, I present leave-one-participant-out analysis (i.e., test with one participant, train with the rest) using both pipelines. The results are calculated using MFCC feature vectors with 26 coefficients, 20 ms window length, and 10 ms overlap.

SVM Pipeline

For the unmodified watch condition using SVM (linear polynomial kernel), the system reports an overall accuracy of 82.1% (sd=6.8%) across 10 participants and 10 events. P7 achieves the highest accuracy at 89.6% and P4 achieves the lowest accuracy at 70.0%. I present the confusion matrix of the results in Figure 6.15. The lowest precision of 66.9% is observed for the clockwise event, mostly confused with a long blow event. Based on the length of the event, it is possible that the two events would be confused with each other. The puff event is once again highly confused with the short blow event and I recommend not including this event in future event sets. The shoosh and sip events are the most accurate at 99.5% and 97.8%, respectively.

For FluteCase using SVM (linear polynomial kernel), leave-one-participant-out analysis across 10 participants and 14 events results in overall accuracy of 84.3% (sd=7.3%). P5 achieves the highest accuracy at 94.6% and P7 achieves the lowest accuracy at 74.5%. The confusion matrix of the results is shown in Figure 6.17. The lowest precision of 67.7% is observed for the top center blow event, mostly confused with a right bezel blow event. I

hypothesize that the orientation and placement of the arm may lead to blows targeted at a particular hole to be directed to adjacent or neighboring holes. For example, as in the case of the right bezel target and the top center. Bringing the watch closer to the mouth may make it easier to aim at particular targets. The bottom center event achieves the highest accuracy at 98.9% which is consistent across other conditions. Other events worth noting are swipe up and swipe down at 92.7% and 93.8%, respectively.

HMM Pipeline

For the unmodified watch condition, the system reports an overall accuracy of 79.9% (sd=11.2%) across 8 participants and 10 events. P7 achieves the highest accuracy at 90.2% and P2 achieves the lowest accuracy at 56.4%. The lowest precision of 58.1% is observed for the long blow event, mostly confused with a short blow event. The shoosh and sip events achieve the highest accuracy at 97.9% and 97.8%, respectively. The shoosh exhibits distinct high frequency energy and the sip is characterized by an inhale action. The confusion matrix of the results is shown in Figure 6.16.

Similarly for FluteCase, preliminary leave-one-participant-out analysis across 8 participants and 14 events results in overall accuracy of 78.6% (sd=6.8%). P5 achieves the highest accuracy at 89.9% and P9 achieves the lowest accuracy at 65.0%. The lowest precision of 64.1% is observed for the top left event, mostly confused with a clockwise event. The bottomcenter event achieves accuracy of 94.7% with the microphone located at the bottom bezel. The two most accurate events are left blow and swipe up at 96.9%. The confusion matrix of the results is shown in Figure 6.18.

Best Case Models Using SVM

I re-evaluated the SVM classification results using the user-independent models with a subset of two, three, and five out of all possible gestures to find the best case models for the commodity watch (10 events total) and FluteCase (14 events total). I calculate all com-

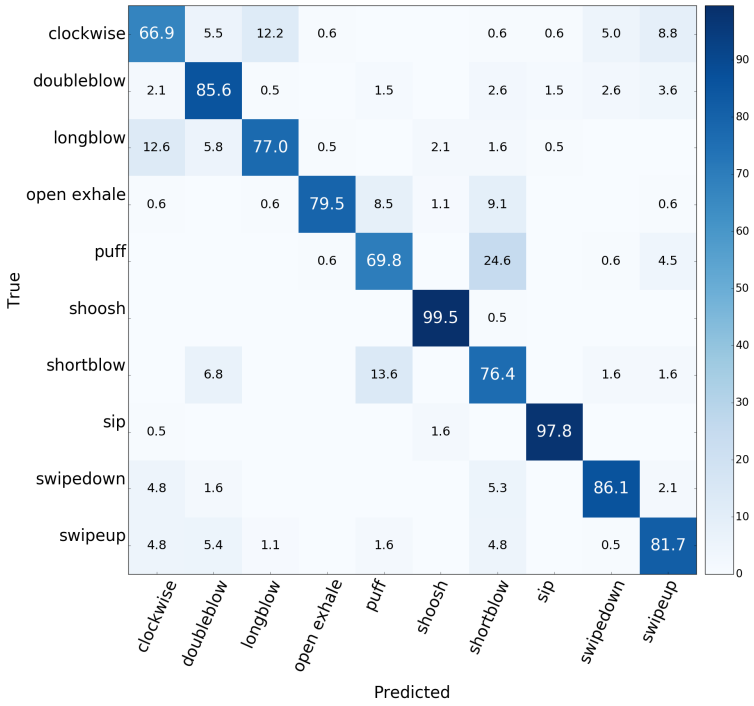


Figure 6.15: Leave-one-out confusion matrix across all users (in %) for the unmodified watch using SVM. Rows represent ground truth and columns are predicted values.

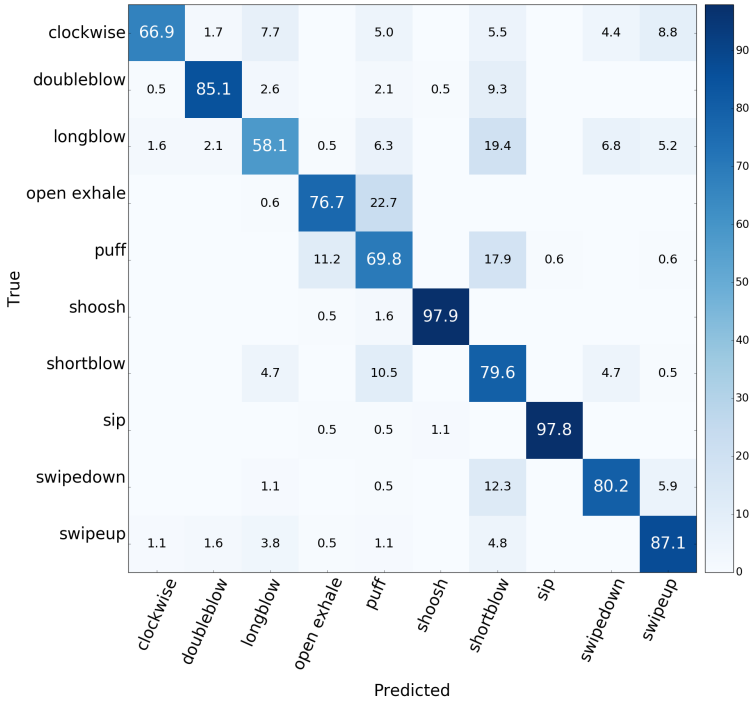


Figure 6.16: Leave-one-out confusion matrix across all users (in %) for the unmodified watch using HMMs. Rows represent ground truth and columns are predicted values.

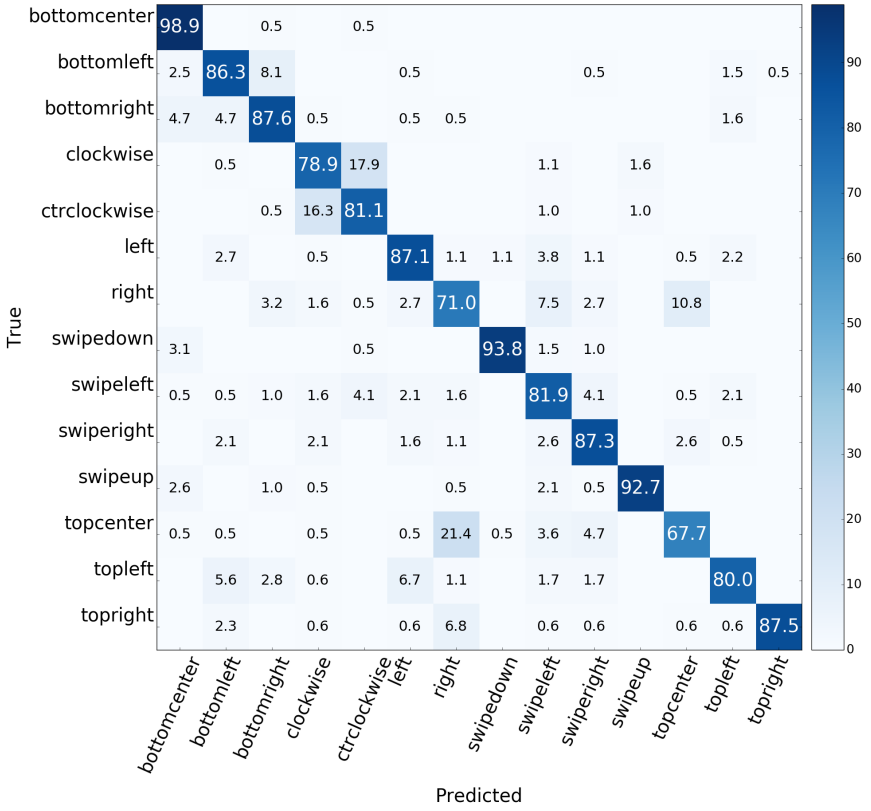


Figure 6.17: Leave-one-out confusion matrix across all users (in %) for FluteCase using SVM. Rows represent ground truth and columns are predicted values.

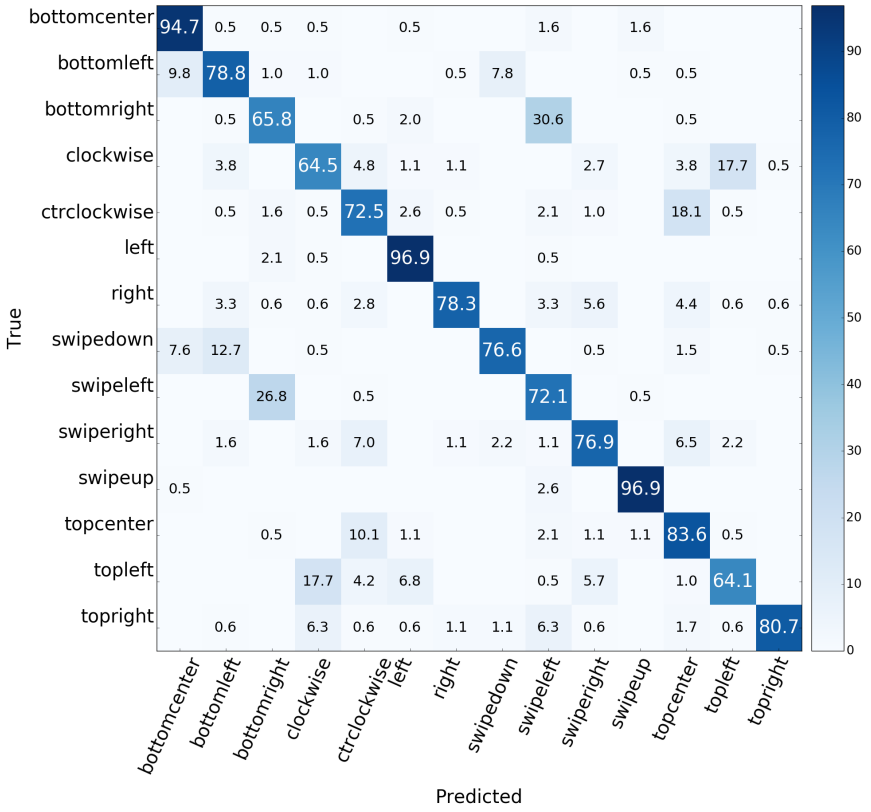


Figure 6.18: Leave-one-out confusion matrix across all users (in %) for FluteCase using HMMs. Rows represent ground truth and columns are predicted values.

binations of the gesture labels (i.e., A choose B) to find a subset and perform leave-one-participant-out validation across all participants with the resulting labels only. In general, there were multiple combinations with comparable accuracies, so I only report the combination of gestures that could most intuitively be used in the design of a control interface.

For two input actions, the commodity watch presents accuracy of 100% for the double blow and shoosh events. The FluteCase results in 100% accuracy for the right bezel blow and counterclockwise events. For three input actions, the commodity watch achieves accuracy of 99.6% (sd=0.4%) for clockwise blow, puff, and shoosh. The FluteCase results achieve 99.4% (sd=1.3%) for a top left bezel blow, left bezel blow, and a clockwise blow.

For five input actions, the commodity watch achieves accuracy of 94.1% (sd=4.8%) for the shoosh, short blow, sip, swipe up, and swipe down events. The combination of these gestures could be used to navigate an interface with swipes and perform selection with a shoosh, short blow, and/or sip. For FluteCase, the system achieves accuracy of 94.3% (sd=4.9%) for swipe right, and bezel blows at the top left, top center, top right, and bottom center.

6.11.6 Discussion

I have presented user-dependent results for the SVM pipeline and user-independent leave-one-participant-out results for the SVM and HMM pipelines. Overall, the per-user accuracy of the system using SVM is 91.8% for the unmodified watch and 92.7% for FluteCase. Results indicate that there is a subset of gestures (e.g., shoosh, short blow, swipe up, swipe down) that would enable a useful set of input actions for the smartwatch. The focus of this work is to demonstrate a proof of concept system and highlight the possibilities of the recognizer with a broad set of acoustic events.

Leave-one-out results across both pipelines are encouraging, yet not as promising as the user dependent results. In general, the SVM pipeline should exhibit better performance with non-directional blow events (e.g., short blow, bezel blows, etc.) and HMMs are ex-

pected to excel with dynamic or directional acoustic events (e.g., air swipes, clockwise, counterclockwise, etc.). A subset of gestures can be used to determine best case models and achieve accuracies above 90% for user-independent usage.

In the unmodified watch condition, the SVM pipeline outperforms HMMs for the following events: long blow and surprisingly swipe down. Most of the energy of the swipe down event is concentrated around the microphone at the bottom center of the bezel, thus appearing like a non-directional event. The HMM outperforms or presents comparable performance in the following events: swipe up, clockwise, short blow. Across the SVM and HMM pipelines, shoosh and sip are the most accurate events. In the FluteCase condition, the SVM pipeline outperforms HMMs for the following events: bottom left, bottom right, long blow, and again swipe down. The HMM outperforms or presents comparable performance in the following events: swipe up, top center, and left bezel blow. Across the SVM and HMM pipelines, shoosh and sip are the most accurate events.

The HMM modeling is based on a single 8-state topology for the entire acoustic event set (unmodified watch and FluteCase). It is possible that by designing custom topologies per condition and per gesture, the overall accuracy of the HMM pipeline would increase and the performance of dynamic gestures would improve. The exercise of further tuning the pipeline and parameters is left for future work.

In general, the results in this chapter demonstrate the promise of Whoosh as an expressive interaction technique for smartwatches. As shown above, a subset of the gestures presented lead to higher accuracies and may be sufficient for a majority of desired input actions on the device.

6.12 Demonstration Applications

I implement several demonstration applications that highlight the potential for Whoosh interactions, with both the unmodified watch and FluteCase. I refer the reader to the video[2]

[2]https://www.youtube.com/watch?v=Hhhchw7PweU

accompanying this work for live demonstrations of each application.

6.12.1 Unmodified Watch Applications

Notifications: Quick access to notifications or quick actions without having to use a mobile phone is arguably one of the most compelling features of a smartwatch. The notifications app shows examples of how Whoosh facilitates such interactions. The app enables *discrete selection* between one or two buttons on-screen. I use the event recognizer to silence or dismiss an incoming call notification with a shoosh event, and answer the call with a single blow acknowledgment.

Authentication: A person can also use a sequence of Whoosh events as an additional layer of security on their devices. The smartwatch can automatically lock whenever the user removes it from the wrist. In the application, a lock screen pops up and a pre-determined sequence of Whoosh events is used to unlock the device. Whoosh events on the watch could also be used as a physical authentication challenge to complete a purchase on another device (e.g., mobile phone or desktop).

Speech + Whoosh: Whoosh events can be combined with speech to create a mixed interaction modality. In the messaging application, a user dictates the content of a text message and uses Whoosh events to manipulate the text. A long blow is used to backspace and a short blow is used to send the message when complete. The user quickly mode switches between speech and Whoosh input using a double blow, or could potentially use a flick of the wrist.

Multi-Device Handoff: When Whoosh is run in parallel on both the smartwatch and phone, it can enable a robust set of multi-device events. For example, I explore interactions between the watch and the phone held in the same hand. Inspired by the stitching technique [49] and Duet [20], I support two multi-device events: watch-to-phone and phone-to-watch *sip-and-puff*. Sip-and-puff provides an intuitive metaphor to transfer tasks from one device to another. In the demonstration, a sip event on the watch "absorbs" content on the

watch screen and a puff event remotely delivers the content to the phone. This allows, for instance, a user who receives an email notification on the watch to transfer and view the entire message on a larger device.

6.12.2 FluteCase Applications

Maps: Whoosh enhances navigation on a map by providing the following actions. Panning the map is enabled by *bezel blows*. In the application, the map shifts in the direction towards the FluteCase hole that the user blows. Continuously blowing into the same hole could keep the map moving in that direction. A total of eight panning directions are enabled with the FluteCase. Zooming is enabled by *circular blows*. In the demonstration, a circular blow in the clockwise direction will zoom in the map while the counterclockwise direction will zoom out. An *air swipe* up or down allows the user to traverse layers of hierarchical content. In the application, a swipe down reveals the various map views (e.g., satellite, terrain) and a swipe up returns up the stack.

Application Shortcuts: Smartwatches are intended to minimize the time between intent and action [6]. In the demonstration, eight app icons are displayed on the watch home screen aligned with the FluteCase holes. The user blows at any of the FluteCase target locations to open the associated app on the watch itself or potentially on the mobile phone.

6.13 Discussion and Future Work

Using audio for interaction can always present potential privacy concerns as the device may capture spoken input from the user. Whoosh focuses on non-speech audio recognition based on extracted features and does not store any raw audio. Furthermore, the Whoosh recognizer is lightweight and runs in real-time on the device. Thus, I do not require sending audio to the cloud for additional computational power and processing.

Whoosh is well-suited as a complementary input modality for smartwatches. A multi-modal approach enables more complex and potentially parallel forms of input. “Chording”

or combining Whoosh events with touch, speech, or motion provides a new set of fluid interactions. One potential example includes "clutch" mechanisms. An air swipe might be used to trigger sending an SMS. A flick of the wrist after the event could cancel, or a touch down during the event could immediately confirm the intended action. Such a mechanism provides a lightweight confirmation step for microinteractions that are irreversible.

The Whoosh recognizer running on a commodity watch without modification enables various simple microinteractions. FluteCase enables an expanded set of interactions but requires modifying the physical structure of the watch. To achieve a richer vocabulary, there is a trade-off between passive approaches such as FluteCase and other solutions at the hardware level. Potential opportunities, which I have not yet explored, include increasing the number of microphones or altering microphone placement.

I have begun exploring the use of Whoosh in-the-wild. The initial evaluation of the activation event with the unmodified watch assesses false positives and negatives. The segmentation is tuned to be robust to noise (as demonstrated in our video figure). For future work, I want to further assess how environmental sounds, such as wind and other unforeseen ambient noise, may potentially impact the system. I could also redesign the FluteCase to minimize the pathway for ambient sounds. Improving the machine learning pipeline could also address these issues. User variability across events is another challenge. Personalization of the classification models to dynamically incorporate new users' data may improve user independent performance.

6.14 Contributions

I present a summary of the Whoosh system and its contributions using this dissertation's research goals. Figure 6.19 presents a summarized version.

Goal 1: Achieving *expressiveness* through a spectrum of interaction events and wearable interfaces

- Type of Interface and Purpose of Interaction

  Whoosh is a non-voice acoustics interface. The system enables rapid microinteractions and can supplement speech for notification-response, command, and activation gestures.

- Number of Unique Input Actions

  A total of 10 and 14 events are supported. A commodity watch supports a total of 10 events with no modifications, and the addition of a 3D-printed watch case provides an additional 14 non-voice acoustic events.

- Information Transfer Rate

  The information transfer rates range between 0.87 to 1.04 bps for the FluteCase and commodity watch, respectively.

- Applications

  For the commodity watch, applications include: multi-device interactions, an authentication app, a D-Pad controller, a music player, and a mixture of speech plus Whoosh use cases. For the FluteCase, applications include: maps navigation and bezel shortcuts.

Goal 2: Enabling rapid *speed* of interaction and reducing the time between intention and action

- Time Between Intention and Action

  The time between intention and action (i.e., setup time) is 1.5 to 2 seconds. Whoosh requires the user to raise the arm and bring it close to the mouth, resulting in up to 2 seconds of setup time.

- Speed of Interaction

  The interaction can be completed within approx. 2 to 3 seconds. Recognition uses a

segmented window with on-device processing suitable for microinteractions. Longer windows are required for more complex events.

Goal 3: Expanding the *practicality* of one-handed input techniques and form factors

- Level of Instrumentation

  Level of instrumentation is low to medium. Acoustic signals can be detected on a commodity watch with no added instrumentation and enhanced with a passive 3D-printed case attached to the watch. No instrumentation of the fingers is needed.

- Prototype Readiness for Deployment

  A 3D-printed watch case can be shrunk and embedded into the watch's bezel or the commodity watch can be used. For the FluteCase, the prototype is rated TRL 4 for a small scale prototype used in the laboratory. For the commodity watch, the system is rated TRL 6 for a prototype system used in the laboratory and in-the-wild.

Goal 4: Building *robust* detection approaches for multiple users and everyday activities

- Accuracy of Technique

  The user-independent accuracy of the technique is reported here for the best case models. Whoosh with the commodity watch achieves 99.6% for a subset of 3 events. The FluteCase achieves 99.4% for a subset of 3 events. More details about the best case models with a subset of events and models with all events are reported in the chapter.

- Usage Across Multiple Users

  The system supports both user-independent and user-dependent models. Classification is robust with individual per-user training. User-independent models with a subset of gestures are stable across users.

- Support During Everyday Activities

  Medium to High support. The technique is robust to ambient noise, in particular for

events that saturate the microphone. Low false positive and negative rates for the double blow event in-the-wild.

6.15 Summary

In this chapter, I present Whoosh which is a sensing technique that uses non-voice acoustic input for microinteractions on smartwatches. The system exploits the unique signature of sounds generated by the user to enable low-cost, hands-free, and rapid input on commodity devices. I evaluated the offline performance of the unmodified watch recognizer with a total of 18 participants and 10 events in two separate lab studies. The recognition system achieves 90.6% and 91.8% user-dependent accuracies in the first and second evaluations respectively using a SVM classifier. The system achieves 71.3%, 82.1% and 79.9% user-independent accuracy using leave-one-participant-out in the first evaluation using SVM, using a SVM in the second evaluation, and using HMMs in the second evaluation, respectively. Using FluteCase, the 3D-printed passive case around the smartwatch, I alter the acoustic response captured by the microphone to enable 14 additional interactions. I evaluated the offline performance of FluteCase with a total of 18 participants and 14 events in two separate lab studies. The recognition system achieves 91.3% and 92.7% user-dependent accuracies in the first and second evaluations using a SVM classifier. The system achieves 79.7%, 84.3% and 78.6% user-independent accuracy using leave-one-participant-out in the first evaluation using SVM, using a SVM in the second evaluation, and using HMMs in the second evaluation, respectively. For live demonstration, the system detects and classifies non-voice acoustic signals in real-time on the device. I conclude with a set of example applications that highlight the technique and demonstrate the design space and opportunities enabled by Whoosh. Whoosh addresses research goals across *expressiveness, speed, practicality* and *robustness* factors, and its contributions are summarized in Figure 6.19.

Dimensions	Commodity Watch	FluteCase
Goal 1: Achieving expressiveness through a spectrum of interaction events and wearable interfaces		
Purpose of Interaction	Command Notification Response Activation	Command Notification Response
Number of unique input actions	10 acoustic events	14 acoustic events
Information transfer rate	1.04 bps	0.87 bps
Applications		
Goal 2: Enabling rapid speed of interaction and reducing the time between intention and action		
Time between intention and action	1.5 to 2 seconds	1.5 to 2 seconds
Speed of interaction	2 seconds	3 seconds
Goal 3: Expanding the practicality of one-handed input techniques and form factors		
Level of instrumentation	Low	Medium
Prototype readiness	TRL 6 Prototype system (laboratory / in-the-wild)	TRL 4 Small scale prototype (laboratory)
Goal 4: Building robust detection approaches for multiple users and everyday activities		
Accuracy of technique	99.6% for 3 events	99.4% for 3 events
Usage across multiple users	User-independent	User-independent
Support during everyday activities	High	Medium

Figure 6.19: Summary of contributions and research goals for Whoosh.

CHAPTER 7
CONCLUSION

Wearable computing devices enable people to access the world's information at a glance. A user can glance at an incoming text message, respond to that message with a voice command, and control their music on-the-go using touch. Touch and speech are the key driving interaction modes for wrist-worn and head-based devices today. However, these input modalities may not be appropriate or desired in many situations where hands are busy or the person is not able to interact due to a physical disability. Input can be enhanced by facilitating use in situations where the watch hand or no hands are needed.

In this dissertation, I describe one-handed input techniques for wearable computing using acoustic and inertial sensing that I have designed, developed, and evaluated. In this chapter, I revisit my thesis, summarize the research goals driving this work, highlight the key contributions of each system, and present final remarks.

7.1 Thesis

In this dissertation, I developed multiple interaction techniques and algorithms using wrist- and finger-worn sensors to enable one-handed input. Specifically, the systems and experiments presented in this dissertation address my thesis statement, which reads as follows:

One-handed input recognized using on-body inertial and acoustic sensing can enable an *expressive*, *fast*, *practical*, and *robust* modality for wearable computing.

To demonstrate my thesis, I highlight four main aspects of wearable computing interactions that I explore across three systems: expressiveness, speed, practicality, and robust-

ness.

In Chapter 4, I explored the broad set of possible thumb and finger gestures by incorporating additional instrumentation of the thumb and wrist using inertial sensing. The direct instrumentation of the thumb enabled a highly expressive set of gestures with over 30 possible gestures, across two hardware iterations. The additional instrumentation results in a less practical but custom-designed solution that could prove to be more comfortable and useful in the future for long term use. Finally, the interaction is robust for per-user classification. However, the large set of gestures and current dataset/features are not as robust for use across different people.

In Chapter 5, I introduce the use of correlation between a person's thumb movement and on-screen stimuli with the first instance of a rhythmic mimicry technique for wrist-worn wearables using sensors in the device. The technique supports binary selection within seconds and is less expressive than the other systems in this dissertation. However, the technique is practical and does not require calibration or machine learning. It is also robust across users and by its nature user-independent.

Lastly, in Chapter 6, I demonstrate that non-voice acoustics can enable a fast, low-cost, and expressive way of providing input to the smartwatch with 24 total acoustic events. The technique is practical for deployment on commodity smartwatches and robust to usage across multiple users and physical settings.

Below I present an overall summary of how each project meets the research goals stated at the beginning of the dissertation.

7.2 Summary of Research Goals

I approached this dissertation through four research goals and using three different interaction techniques as case studies. The research goals of this dissertation are:

- Goal 1: Achieving *expressiveness* through a spectrum of interaction events and wearable interfaces

- Goal 2: Enabling rapid *speed* of interaction and reducing the time between intention and action
- Goal 3: Expanding the *practicality* of one-handed input techniques and form factors
- Goal 4: Building *robust* detection approaches for multiple users and everyday activities

I frame and characterize the systems in this dissertation using the research goals below:

Goal 1: Achieving *expressiveness* through a spectrum of interaction events and wearable interfaces

- Type of Interface and Purpose of Interaction
  - Thumby: Finger-Level Motion Gesture Interface

    Inertial sensing of the wrist and thumb eliminates smartwatch occlusion and provides a broad set of notification-response and command gestures.
  - SynchroWatch: Synchronous Gesture Interface

    SynchroWatch is a synchronous gesture interface. The system uses rhythmic mimicry of thumb movements to on-screen stimuli which can be detected within 3 seconds and used for notification-response and command gestures.
  - Whoosh: Non-Voice Acoustics Interface

    The system enables rapid microinteractions and can supplement speech for notification-response, command, and activation gestures.
- Number of Unique Input Actions
  - Thumby: Total of 37 unique gestures using separate models

    The first iteration of Thumby supports up to 12 thumb-to-phalange pinches, 8

drawing thumb gestures on the palm of the hand, and 9 taps/swipes/flick gestures. The second iteration supports 8 gestures including taps, swipes, and other thumb-and-wrist gestures.

  - SynchroWatch: 2 to 4 events

    The technique is designed to support binary selection of two blinking targets. A separate flick of the wrist gesture can be used to alternate between two pairs of targets for selection of up to 4 events.

  - Whoosh: 10 and 14 events

    A total of 10 or 14 events are supported. A commodity watch supports a total of 10 events with no modifications, and the addition of a 3D-printed watch case provides an additional 14 non-voice acoustic events.

- Information Transfer Rate (ITR)

  The ITR [80] is dependent on the accuracy of the technique, the number of total input actions supported, and the interactions per second.

  - Thumby:

    The information transfer rates range between 0.69 to 1.15 bps for Thumby v1 and Thumby v2, respectively.

  - SynchroWatch:

    The information transfer rate is 0.18 bps. The value is relatively lower than other systems based on the number of input actions supported (i.e., only binary selection).

  - Whoosh:

    The information transfer rates range between 0.87 to 1.04 bps for the FluteCase and commodity watch, respectively.

- Demonstration Applications

- Thumby:

  For Thumby v1, I have developed applications include controlling music on a smartwatch, a navigational controller for Google Glass, and alphanumerical input app using T9. Other envisioned applications are detailed for smartwatch input with Thumby v2.

- SynchroWatch:

  Smartwatch applications I have developed include demonstrations of answering and dismissing a phone call, scrolling through text or UI cards, and controlling your music player.

- Whoosh:

  For the commodity watch, the applications I have built include multi-device interactions, an authentication app, a D-Pad controller, a music player, and a mixture of speech plus Whoosh use cases. For the FluteCase, applications include maps navigation and bezel shortcuts.

Goal 2: Enabling rapid *speed* of interaction and reducing the time between intention and action

- Time Between Intention and Action

  - Thumby: Approx. 2 seconds

    Recognition is performed using a segmented window of sensor data and analysis off-device, both offline and online.

  - SynchroWatch: 1.5 to 2 seconds

    SynchroWatch is slightly slower than touch swiping but only requires one hand.

  - Whoosh: Approx. up to 2 seconds

    The time between intention and action (i.e., setup time) is 1.5 to 2 seconds. Whoosh requires the user to raise the arm and bring it close to the mouth, resulting in up to 2 seconds of setup time.

- Speed of Interaction
  - Thumby: Approx. 2 seconds

    Recognition is performed using a segmented window of sensor data and analysis off-device, both offline and online.
  - SynchroWatch: 3 seconds or more

    The interaction can be completed within 3 seconds or more. A correlation-based technique with a threshold and user performance determines the time to trigger the binary selection. SynchroWatch is slightly slower than touch swiping but only requires one hand.
  - Whoosh: Approx. 2 to 3 seconds

    The interaction can be completed within approx. 2 to 3 seconds. Recognition uses a segmented window with on-device processing suitable for microinteractions. Longer windows are required for more complex events.

Goal 3: Expanding the *practicality* of one-handed input techniques and form factors

- Level of Instrumentation

  The level of instrumentation is a subjective measure of the amount and type of hardware required to enable the interaction technique, ranging from low to high. The measure is based on the current state of the prototype, not necessarily the final design.
  - Thumby: High

    An active inertial sensor is worn on the thumb and paired with a smartwatch to capture the absolute orientation and movement of the thumb and wrist.
  - SynchroWatch: Medium

    Level of instrumentation is medium. Rhythmic thumb movement can be captured using only a passive magnetic ring worn on the thumb, which alters the

magnetic field around the smartwatch. No battery power or wireless connectivity is necessary.

  - Whoosh: Low to Medium

    Level of instrumentation is low to medium. Acoustic signals can be detected on a commodity watch with no added instrumentation and enhanced with a passive 3D-printed case attached to the watch. No instrumentation of the fingers is needed.

- Prototype Readiness for Deployment

  - Thumby: TRL 4 and TRL 5

    The wired thumb ring (Thumby v1) requires an external laptop for processing (TRL 4 for a small scale prototype used in the laboratory). The battery-powered and wireless inertial thumb ring (Thumby v2) pairs with a smartwatch in a slightly bulky form factor (TRL 5 for a large scale prototype used in the intended environment).

  - SynchroWatch: TRL 5

    The prototype readiness for deployment is rated TRL 5, which corresponds to a large scale prototype used in its intended environment. A 3D-printed case houses a neodymium magnet and could be designed as a fashionable accessory in the future.

  - Whoosh: TRL 4 and TRL 6

    A 3D-printed watch case can be shrunk and embedded into the watch's bezel or the commodity watch can be used. For the FluteCase, the prototype is rated TRL 4 for a small scale prototype used in the laboratory. For the commodity watch, the system is rated TRL 6 for a prototype system used in the laboratory and in-the-wild.

Goal 4: Building *robust* detection approaches for multiple users and everyday activities

- Accuracy of Technique
  - Thumby: 90.1% for 4 pinches & 90.7% for thumb gestures

    Thumb-to-fingertip pinches and thumb gestures with thumb- and wrist-worn sensors achieve the highest interactive accuracy rates. Other interactions and accuracies are described in Chapter 4.
  - SynchroWatch: 85% accuracy with correlation threshold within 3 seconds

    Detection of thumb synchronous is correlation dependent. See full details in Chapter 5.
  - Whoosh: 99.4% to 99.6% for best case models

    The user-independent accuracy of the technique is reported here for the best case models. Whoosh with the commodity watch achieves 99.6% for a subset of 3 events. The FluteCase achieves 99.4% for a subset of 3 events. More details about the best case models with a subset of events and models with all events are reported in Chapter 6.
- Usage Across Multiple Users
  - Thumby: User-dependent

    The system is currently user-dependent. Classification is robust with individual per-user training. User-independent models are not stable across users (< 30% accuracy) with the current data set and features used.
  - SynchroWatch: User-independent

    SynchroWatch is a user-independent system. Detection of thumb synchronous correlation offline and online is dependent on correlation coefficient thresholds that are user-independent.
  - Whoosh: User-independent

    The system supports both user-independent and user-dependent models. Clas-

sification is robust with individual per-user training. User-independent models with a subset of gestures are stable across users.

- Support During Everyday Activities
  - Thumby: Medium

    Classification is robust to various levels of global motion during sitting and walking, using a combination of wrist- and thumb-worn inertial sensors. Thumby v1 results are reported using an online version of the system. Thumby v2 results are based on offline analysis and further work is required to build an online system.
  - SynchroWatch: High

    Detection of thumb synchronous correlation offline and online is robust during walking, browsing, and sitting activities. The detection is only supported when the arm is raised and interface is in the user's field of view.
  - Whoosh: Medium to High

    The technique is robust to ambient noise, in particular for events that saturate the microphone. Low false positive and negative rates for the double blow event in-the-wild. High support for the commodity watch and medium for the FluteCase.

7.3 Final Remarks

Wearables bring computing closer to the senses and enable a unique experience that seeks to provide information at a glance, reduce the time between intent and action, and enable new types of on-body interaction and sensing opportunities. I presented three interaction techniques that facilitate input for microinteractions on wrist-worn devices using acoustic and inertial sensing. These interactions lie on a spectrum of input possibilities from binary to multi-target selection, using varying levels of instrumentation of the watch and fingers,

along with both machine learning and non-ML techniques. As on-body computation and sensing continues to improve, the opportunities to support new input techniques and situations will continue to evolve. This dissertation provides a set of interaction techniques and a framework for designing and evaluating new modalities that future wearable computing researchers and designers may build upon.

REFERENCES

[1] G. D. Abowd, "What next, ubicomp?: Celebrating an intellectual disappearing act", in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12, New York, NY, USA: ACM, 2012, pp. 31–40.

[2] B. Amento, W. Hill, and L. Terveen, "The Sound of One Hand: A Wrist-mounted Bio-acoustic Fingertip Gesture Interface", in *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '02, New York, NY, USA: ACM, 2002, pp. 724–725.

[3] S. Aoyama, B. Shizuki, and J. Tanaka, "ThumbSlide: An Interaction Technique for Smartwatches Using a Thumb Slide Movement", in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '16, New York, NY, USA: ACM, 2016, pp. 2403–2409.

[4] D. Ashbrook, P. Baudisch, and S. White, "Nenya: Subtle and Eyes-free Mobile Input with a Magnetically-tracked Finger Ring", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11, New York, NY, USA: ACM, 2011, pp. 2043–2046.

[5] D. Ashbrook and T. Starner, "MAGIC: A Motion Gesture Design Tool", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10, New York, NY, USA: ACM, 2010, pp. 2159–2168.

[6] D. L. Ashbrook, "Enabling Mobile Microinteractions", AAI3414437, PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2010.

[7] D. L. Ashbrook, J. R. Clawson, K. Lyons, T. E. Starner, and N. Patel, "Quickdraw: The Impact of Mobility and On-body Placement on Device Access Time", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08, New York, NY, USA: ACM, 2008, pp. 219–222.

[8] P. Baudisch and G. Chu, "Back-of-device interaction allows creating very small touch devices", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09, New York, NY, USA: ACM, 2009, pp. 1923–1932.

[9] R. S. Boyer and J. S. Moore, "MJRTYâăť A Fast Majority Vote Algorithm", in *Automated Reasoning*, ser. Automated Reasoning Series 1, R. S. Boyer, Ed., DOI: 10.1007/978-94-011-3488-0_5, Springer Netherlands, 1991, pp. 105–117.

[10] A. Brajdic and R. Harle, "Walk Detection and Step Counting on Unconstrained Smartphones", in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '13, New York, NY, USA: ACM, 2013, pp. 225–234.

[11] L. Buechley and M. Eisenberg, "Fabric PCBs, electronic sequins, and socket buttons: Techniques for e-textile craft", *Personal and Ubiquitous Computing*, vol. 13, no. 2, pp. 133–150, Feb. 2009.

[12] A. Bulling, U. Blanke, and B. Schiele, "A Tutorial on Human Activity Recognition Using Body-worn Inertial Sensors", *ACM Comput. Surv.*, vol. 46, no. 3, 33:1–33:33, Jan. 2014.

[13] W. Buxton, "There's More to Interaction than Meets the Eye: Some Issues in Manual Input", *User Centered System Design: New Perspectives on Human-Computer Interaction*, pp. 319–337, 1986.

[14] W. Buxton, "Lexical and Pragmatic Considerations of Input Structures", *SIGGRAPH Comput. Graph.*, vol. 17, no. 1, pp. 31–37, Jan. 1983.

[15] A. Cao, K. K. Chintamani, A. K. Pandya, and R. D. Ellis, "NASA TLX: Software for assessing subjective mental workload", *Behavior research methods*, vol. 41, no. 1, pp. 113–117, 2009.

[16] S. K. Card, J. D. Mackinlay, and G. G. Robertson, "The design space of input devices", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '90, New York, NY, USA: ACM, 1990, pp. 117–124.

[17] M. Carter, E. Velloso, J. Downs, A. Sellen, K. O'Hara, and F. Vetere, "PathSync: Multi-User Gestural Interaction with Touchless Rhythmic Path Mimicry", in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16, New York, NY, USA: ACM, 2016, pp. 3415–3427.

[18] K.-Y. Chen, K. Lyons, S. White, and S. Patel, "uTrack: 3d Input Using Two Magnetic Sensors", in *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '13, New York, NY, USA: ACM, 2013, pp. 237–244.

[19] W.-H. Chen, "Blowatch: Blowable and Hands-Free Interaction for Smartwatches", in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '15, New York, NY, USA: ACM, 2015, pp. 103–108.

[20] X. A. Chen, T. Grossman, D. J. Wigdor, and G. Fitzmaurice, "Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch", in *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, ser. CHI '14, New York, NY, USA: ACM, 2014, pp. 159–168.

[21] C. Clarke, A. Bellino, A. Esteves, E. Velloso, and H. Gellersen, "TraceMatch: A Computer Vision Technique for User Input by Tracing of Animated Controls", in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '16, New York, NY, USA: ACM, 2016, pp. 298–303.

[22] A. Dementyev and J. A. Paradiso, "WristFlex: Low-power Gesture Input with Wrist-worn Pressure Sensors", in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '14, New York, NY, USA: ACM, 2014, pp. 161–166.

[23] T. Deyle, S. Palinko, E. S. Poole, and T. Starner, "Hambone: A Bio-Acoustic Gesture Interface", in *2007 11th IEEE International Symposium on Wearable Computers*, ser. ISWC '07, Oct. 2007, pp. 3–10.

[24] M. Dhuliawala, J. Lee, J. Shimizu, A. Bulling, K. Kunze, T. Starner, and W. Woo, "Smooth Eye Movement Interaction Using EOG Glasses", in *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ser. ICMI 2016, New York, NY, USA: ACM, 2016, pp. 307–311.

[25] A. Esteves, E. Velloso, A. Bulling, and H. Gellersen, "Orbits: Gaze Interaction for Smart Watches Using Smooth Pursuit Eye Movements", in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ser. UIST '15, New York, NY, USA: ACM, 2015, pp. 457–466.

[26] G. E. Fabiani, D. J. McFarland, J. R. Wolpaw, and G. Pfurtscheller, "Conversion of EEG activity into cursor movement by a brain-computer interface (BCI)", *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 12, no. 3, pp. 331–338, Sep. 2004.

[27] L. Fehr, W. E. Langbein, and S. B. Skaar, "Adequacy of Power Wheelchair Control Interfaces for Persons with Severe Disabilities: A Clinical Survey", *J. of Rehab R&D '00*, pp. 353–360, 2000.

[28] J. F. Filho, W. Prata, and T. Valle, "Advances on Breathing Based Text Input for Mobile Devices", in *Universal Access in Human-Computer Interaction '15*, DOI: 10.1007/978-3-319-20678-3_27, 2015, pp. 279–287.

[29] R. Fukui, M. Watanabe, T. Gyota, M. Shimosaka, and T. Sato, "Hand Shape Classification with a Wrist Contour Sensor: Development of a Prototype Device", in *Proceedings of the 13th International Conference on Ubiquitous Computing*, ser. UbiComp '11, New York, NY, USA: ACM, 2011, pp. 311–314.

[30] M. Goldstein and D. Chincholle, "Finger-joint gesture wearable keypad", in *Second Workshop on Human Computer Interaction with Mobile Devices*, 1999.

[31] J. Gong, X.-D. Yang, and P. Irani, "WristWhirl: One-handed Continuous Smartwatch Input Using Wrist Gestures", in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ser. UIST '16, New York, NY, USA: ACM, 2016, pp. 861–872.

[32] M. Gordon, T. Ouyang, and S. Zhai, "WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding", in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16, New York, NY, USA: ACM, 2016, pp. 3817–3821.

[33] T. Gotzelmann and P.-P. Vazquez, "InclineType: An Accelerometer-based Typing Approach for Smartwatches", in *Proceedings of the XVI International Conference on Human Computer Interaction*, ser. Interaccion '15, New York, NY, USA: ACM, 2015, 59:1–59:4.

[34] D. Grimes, D. S. Tan, S. E. Hudson, P. Shenoy, and R. P. Rao, "Feasibility and Pragmatics of Classifying Working Memory Load with an Electroencephalograph", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08, New York, NY, USA: ACM, 2008, pp. 835–844.

[35] A. Guo and T. Paek, "Exploring Tilt for No-touch, Wrist-only Interactions on Smartwatches", in *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, ser. MobileHCI '16, New York, NY, USA: ACM, 2016, pp. 17–28.

[36] N. Y. Hammerla, R. Kirkham, P. Andras, and T. Ploetz, "On Preserving Statistical Characteristics of Accelerometry Data Using Their Empirical Cumulative Distribution", in *Proceedings of the 2013 International Symposium on Wearable Computers*, ser. ISWC '13, New York, NY, USA: ACM, 2013, pp. 65–68.

[37] S. Harada, J. A. Landay, J. Malkin, X. Li, and J. A. Bilmes, "The Vocal Joystick:: Evaluation of Voice-based Cursor Control Techniques", in *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, ser. Assets '06, New York, NY, USA: ACM, 2006, pp. 197–204.

[38] C. Harrison and S. Hudson, "Scratch input: Creating large, inexpensive, unpowered and mobile finger input surfaces", in *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ser. UIST '08, New York, NY, USA: ACM, 2008, pp. 205–208.

[39] C. Harrison, H. Benko, and A. D. Wilson, "OmniTouch: Wearable Multitouch Interaction Everywhere", in *Proceedings of the 24th Annual ACM Symposium on User*

Interface Software and Technology, ser. UIST '11, New York, NY, USA: ACM, 2011, pp. 441–450.

[40] C. Harrison and S. E. Hudson, "Abracadabra: Wireless, High-precision, and Unpowered Finger Input for Very Small Mobile Devices", in *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '09, New York, NY, USA: ACM, 2009, pp. 121–124.

[41] C. Harrison, J. Schwarz, and S. E. Hudson, "TapSense: Enhancing finger interaction on touch surfaces", in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ser. UIST '11, New York, NY, USA: ACM, 2011, pp. 627–636.

[42] C. Harrison, D. Tan, and D. Morris, "Skinput: Appropriating the Body as an Input Surface", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10, New York, NY, USA: ACM, 2010, pp. 453–462.

[43] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research", *Advances in psychology*, vol. 52, pp. 139–183, 1988.

[44] R. Heise and B. A. MacDonald, "Quaternions and Motion Interpolation: A Tutorial", in *New Advances in Computer Graphics*, DOI: 10.1007/978-4-431-68093-2_14, Springer, Tokyo, 1989, pp. 229–243.

[45] W. Higgins, "A Comparison of Complementary and Kalman Filtering", *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-11, no. 3, pp. 321–325, May 1975.

[46] S. G. Hill, H. P. Iavecchia, J. C. Byers, A. C. Bittner Jr, A. L. Zaklade, and R. E. Christ, "Comparison of four subjective workload rating scales", *Human factors*, vol. 34, no. 4, pp. 429–439, 1992.

[47] K. Hinckley, "Synchronous Gestures for Multiple Persons and Computers", in *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '03, New York, NY, USA: ACM, 2003, pp. 149–158.

[48] K. Hinckley, R. Pausch, and D. Proffitt, "Attention and Visual Feedback: The Bimanual Frame of Reference", in *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, ser. I3D '97, New York, NY, USA: ACM, 1997, 121–ff.

[49] K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch, and M. Smith, "Stitching: Pen Gestures That Span Multiple Displays", in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '04, New York, NY, USA: ACM, 2004, pp. 23–31.

[50] L. M. Hirshfield, E. T. Solovey, A. Girouard, J. Kebinger, R. J. Jacob, A. Sassaroli, and S. Fantini, "Brain Measurement for Usability Testing and Adaptive Interfaces: An Example of Uncovering Syntactic Workload with Functional Near Infrared Spectroscopy", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09, New York, NY, USA: ACM, 2009, pp. 2185–2194.

[51] C.-E. Hrabia, K. Wolf, and M. Wilhelm, "Whole Hand Modeling Using 8 Wearable Sensors: Biomechanics for Hand Pose Prediction", in *Proceedings of the 4th Augmented Human International Conference*, ser. AH '13, New York, NY, USA: ACM, 2013, pp. 21–28.

[52] D.-Y. Huang, L. Chan, S. Yang, F. Wang, R.-H. Liang, D.-N. Yang, Y.-P. Hung, and B.-Y. Chen, "DigitSpace: Designing Thumb-to-Fingers Touch Interfaces for One-Handed and Eyes-Free Interactions", in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16, New York, NY, USA: ACM, 2016, pp. 1526–1537.

[53] S. E. Hudson, C. Harrison, B. L. Harrison, and A. LaMarca, "Whack gestures: Inexact and inattentive interaction with mobile devices", in *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, ser. TEI '10, New York, NY, USA: ACM, 2010, pp. 109–112.

[54] X. Huo and M. Ghovanloo, "Using Unconstrained Tongue Motion as an Alternative Control Mechanism for Wheeled Mobility", *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 6, pp. 1719–1726, 2009.

[55] T. Igarashi and J. F. Hughes, "Voice As Sound: Using Non-verbal Voice Input for Interactive Control", in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '01, New York, NY, USA: ACM, 2001, pp. 155–156.

[56] A. K. Karlson and B. B. Bederson, "ThumbSpace: Generalized One-Handed Input for Touchscreen-Based Mobile Devices", in *INTERACT '07*, DOI: 10.1007/978-3-540-74796-3_30, 2007, pp. 324–338.

[57] J. Kela, P. Korpipaa, J. Mantyjarvi, S. Kallio, G. Savino, L. Jozzo, and D. Marca, "Accelerometer-based Gesture Control for a Design Environment", *Personal Ubiquitous Comput.*, vol. 10, no. 5, pp. 285–299, Jul. 2006.

[58] F. Kerber, P. Lessel, and A. Kruger, "Same-side Hand Interactions with Arm-placed Devices Using EMG", in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '15, New York, NY, USA: ACM, 2015, pp. 1367–1372.

[59] W. Kienzle and K. Hinckley, "LightRing: Always-available 2d Input on Any Surface", in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '14, New York, NY, USA: ACM, 2014, pp. 157–160.

[60] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, J. Chen, I. Oikonomidis, and P. Olivier, "Digits: Freehand 3d Interactions Anywhere Using a Wrist-worn Gloveless Sensor", in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '12, New York, NY, USA: ACM, 2012, pp. 167–176.

[61] J.-W. Kim, H.-J. Kim, and T.-J. Nam, "M.Gesture: An Acceleration-Based Gesture Authoring System on Multiple Handheld and Wearable Devices", in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16, New York, NY, USA: ACM, 2016, pp. 2307–2318.

[62] G. Laput, E. Brockmeyer, S. E. Hudson, and C. Harrison, "Acoustruments: Passive, Acoustically-Driven, Interactive Controls for Handheld Devices", in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15, New York, NY, USA: ACM, 2015, pp. 2161–2170.

[63] G. Laput, R. Xiao, X. A. Chen, S. E. Hudson, and C. Harrison, "Skin Buttons: Cheap, Small, Low-powered and Clickable Fixed-icon Laser Projectors", in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '14, New York, NY, USA: ACM, 2014, pp. 389–394.

[64] S. C. Lee and T. Starner, "BuzzWear: Alert Perception in Wearable Tactile Displays on the Wrist", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10, New York, NY, USA: ACM, 2010, pp. 433–442.

[65] F. C. Y. Li, R. T. Guy, K. Yatani, and K. N. Truong, "The 1line Keyboard: A QWERTY Layout in a Single Line", in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '11, New York, NY, USA: ACM, 2011, pp. 461–470.

[66] J.-W. Lin, C. Wang, Y. Y. Huang, K.-T. Chou, H.-Y. Chen, W.-L. Tseng, and M. Y. Chen, "BackHand: Sensing Hand Gestures via Back of the Hand", in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ser. UIST '15, New York, NY, USA: ACM, 2015, pp. 557–564.

[67] C. Loclair, S. Gustafson, and P. Baudisch, "PinchWatch: A wearable device for one-handed microinteractions", in *Proc. MobileHCI*, vol. 10, 2010.

[68] Z. Lu, X. Chen, Z. Zhao, and K. Wang, "A Prototype of Gesture-based Interface", in *Proceedings of the 13th International Conference on Human Computer Interac-*

tion with Mobile Devices and Services, ser. MobileHCI '11, New York, NY, USA: ACM, 2011, pp. 33–36.

[69] K. Lyons, D. Nguyen, D. Ashbrook, and S. White, "Facet: A Multi-segment Wrist Worn System", in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '12, New York, NY, USA: ACM, 2012, pp. 123–130.

[70] K. Lyons, T. Starner, and B. Gane, "Experimental evaluations of the twiddler one-handed chording mobile keyboard", *Human-Computer Interaction*, vol. 21, no. 4, pp. 343–392, 2006.

[71] K. Lyons, T. Starner, D. Plaisted, J. Fusia, A. Lyons, A. Drew, and E. W. Looney, "Twiddler Typing: One-handed Chording Text Entry for Mobile Phones", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '04, New York, NY, USA: ACM, 2004, pp. 671–678.

[72] T. Mai, *NASA Technology Readiness Levels (TRL)*, May 2015.

[73] R. L. Mandryk and M. S. Atkins, "A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies", *International Journal of Human-Computer Studies*, Evaluating affective interactions, vol. 65, no. 4, pp. 329–347, Apr. 2007.

[74] S. Mann, "Wearable computing: A first step toward personal imaging", *Computer*, vol. 30, no. 2, pp. 25–32, Feb. 1997.

[75] P. Mistry, P. Maes, and L. Chang, "WUW - wear Ur world: A wearable gestural interface", in *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '09, New York, NY, USA: ACM, 2009, pp. 4111–4116.

[76] D. Morris, T. S. Saponas, and D. Tan, "Emerging Input Technologies for Always-Available Mobile Interaction", *Found. Trends Hum.-Comput. Interact.*, vol. 4, no. 4, pp. 245–316, Apr. 2011.

[77] S. Nanayakkara, R. Shilkrot, K. P. Yeo, and P. Maes, "EyeRing: A finger-worn input device for seamless interactions with our surroundings", in *Proceedings of the 4th Augmented Human International Conference*, ser. AH '13, New York, NY, USA: ACM, 2013, pp. 13–20.

[78] A. W. Y. Ng and A. H. S. Chan, "Finger Response Times to Visual, Auditory and Tactile Modality Stimuli", *Lecture Notes in Engineering and Computer Science*, vol. 2196, no. 1, pp. 1449–1454, Mar. 2012.

[79] I. Oakley and D. Lee, “Interaction on the Edge: Offset Sensing for Small Devices”, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’14, New York, NY, USA: ACM, 2014, pp. 169–178.

[80] B. Obermaier, C. Neuper, C. Guger, and G. Pfurtscheller, “Information transfer rate in a five-classes brain-computer interface”, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 9, no. 3, pp. 283–288, Sep. 2001.

[81] M. Ogata, Y. Sugiura, H. Osawa, and M. Imai, “iRing: Intelligent Ring Using Infrared Reflection”, in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST ’12, New York, NY, USA: ACM, 2012, pp. 131–136.

[82] A. Olwal and S. Feiner, “Interaction Techniques Using Prosodic Features of Speech and Audio Localization”, in *Proceedings of the 10th International Conference on Intelligent User Interfaces*, ser. IUI ’05, New York, NY, USA: ACM, 2005, pp. 284–286.

[83] S. Park, K. Mackenzie, and S. Jayaraman, “The Wearable Motherboard: A Framework for Personalized Mobile Information Processing (PMIP)”, in *Proceedings of the 39th Annual Design Automation Conference*, ser. DAC ’02, New York, NY, USA: ACM, 2002, pp. 170–174.

[84] A. Parnami, A. Gupta, G. Reyes, R. Sadana, Y. Li, and G. Abowd, “Mogeste: Mobile Tool for In-situ Motion Gesture Design”, in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: ADJUNCT*, ser. UbiComp ’16, New York, NY, USA: ACM, 2016, pp. 345–348.

[85] S. N. Patel, “Infrastructure mediated sensing”, AAI3327635, PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2008.

[86] S. N. Patel and G. D. Abowd, “BLUI: Low-Cost Localized Blowable User Interfaces”, in *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST ’07, New York, NY, USA: ACM, 2007, pp. 217–220.

[87] J. K. Perng, B. Fisher, S. Hollar, and K. S. J. Pister, “Acceleration Sensing Glove”, in *2012 16th International Symposium on Wearable Computers*, vol. 0, Los Alamitos, CA, USA: IEEE Computer Society, 1999, p. 178.

[88] S. T. Perrault, E. Lecolinet, J. Eagan, and Y. Guiard, “WatchIt: Simple Gestures and Eyes-free Interaction for Wristwatches and Bracelets”, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’13, New York, NY, USA: ACM, 2013, pp. 1451–1460.

[89] E. R. Post and M. Orth, “Smart Fabric, or "Wearable Clothing"”, in *Iswc*, IEEE, 1997, p. 167.

[90] I. Poupyrev, N.-W. Gong, S. Fukuhara, M. E. Karagozler, C. Schwesig, and K. E. Robinson, “Project Jacquard: Interactive Digital Textiles at Scale”, in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’16, New York, NY, USA: ACM, 2016, pp. 4216–4227.

[91] V. R. Pratt, “Thumbcode: A Device-Independent Digital Sign Language”, in *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science, Brunswick, NJ*, 1998.

[92] J. Rekimoto, “GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices”, in *Fifth International Symposium on Wearable Computers, 2001. Proceedings*, ser. ISWC ’01, 2001, pp. 21–27.

[93] G. Reyes, D. Zhang, S. Ghosh, P. Shah, J. Wu, A. Parnami, B. Bercik, T. Starner, G. D. Abowd, and W. K. Edwards, “Whoosh: Non-voice Acoustics for Low-cost, Hands-free, and Rapid Input on Smartwatches”, in *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, ser. ISWC 2016, New York, NY, USA: ACM, 2016, pp. 120–127.

[94] B. J. Rhodes and P. Maes, “Just-in-time information retrieval agents”, *IBM Systems Journal*, vol. 39, no. 3.4, pp. 685–704, 2000.

[95] J. Ruiz and Y. Li, “DoubleFlip: A Motion Gesture Delimiter for Mobile Interaction”, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’11, New York, NY, USA: ACM, 2011, pp. 2717–2720.

[96] E. Saeedi, S. Kim, and B. A. Parviz, “Self-assembled crystalline semiconductor optoelectronics on glass and plastic”, *Journal of Micromechanics and Microengineering*, vol. 18, no. 7, p. 075 019, 2008.

[97] D. Sakamoto, T. Komatsu, and T. Igarashi, “Voice Augmented Manipulation: Using Paralinguistic Information to Manipulate Mobile Devices”, in *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, ser. MobileHCI ’13, New York, NY, USA: ACM, 2013, pp. 69–78.

[98] T. S. Saponas, D. Kelly, B. A. Parviz, and D. S. Tan, “Optically sensing tongue gestures for computer input”, in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ser. UIST ’09, New York, NY, USA: ACM, 2009, pp. 177–180.

[99] T. S. Saponas, D. S. Tan, D. Morris, R. Balakrishnan, J. Turner, and J. A. Landay, "Enabling Always-Available Input with Muscle-Computer Interfaces", in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ser. UIST '09, New York, NY, USA: ACM, 2009, pp. 167–176.

[100] N. Sawhney and C. Schmandt, "Nomadic Radio: Scaleable and Contextual Notification for Wearable Audio Messaging", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '99, New York, NY, USA: ACM, 1999, pp. 96–103.

[101] K. A. Siek, Y. Rogers, and K. H. Connelly, "Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs", in *Human-Computer Interaction - INTERACT 2005*, DOI: 10.1007/11555261_24, Springer, Berlin, Heidelberg, Sep. 2005, pp. 267–280.

[102] A. J. Sporka, S. H. Kurniawan, M. Mahmud, and P. Slavik, "Non-speech Input and Speech Recognition for Real-time Control of Computer Games", in *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, ser. Assets '06, New York, NY, USA: ACM, 2006, pp. 213–220.

[103] T. Starner, "Project Glass: An Extension of the Self", *IEEE Pervasive Computing*, vol. 12, no. 2, pp. 14–16, Apr. 2013.

[104] ——, "The challenges of wearable computing: Part 1", *IEEE Micro*, vol. 21, no. 4, pp. 44–52, Jul. 2001.

[105] ——, "The challenges of wearable computing: Part 2", *IEEE Micro*, vol. 21, no. 4, pp. 54–67, Jul. 2001.

[106] T. Starner, J. Auxier, D. Ashbrook, and M. Gandy, "The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring", in *The Fourth International Symposium on Wearable Computers*, Oct. 2000, pp. 87–94.

[107] D. J. Sturman and D. Zeltzer, "A Survey of Glove-based Input", *IEEE Comput. Graph. Appl.*, vol. 14, no. 1, pp. 30–39, Jan. 1994.

[108] E. Thomaz, V. Bettadapura, G. Reyes, M. Sandesh, G. Schindler, T. Ploetz, G. D. Abowd, and I. Essa, "Recognizing water-based activities in the home through infrastructure-mediated sensing", in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12, New York, NY, USA: ACM, 2012, pp. 85–94.

[109] H.-R. Tsai, C.-Y. Wu, L.-T. Huang, and Y.-P. Hung, "ThumbRing: Private Interactions Using One-handed Thumb Motion Input on Finger Segments", in *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile*

Devices and Services Adjunct, ser. MobileHCI '16, New York, NY, USA: ACM, 2016, pp. 791–798.

[110] K. Tsukada and M. Yasumura, *Ubi-Finger: Gesture Input Device for Mobile Use*. 2002.

[111] A. Vardy, J. Robinson, and L.-T. Cheng, "The WristCam as input device", in *The Third International Symposium on Wearable Computers, 1999. Digest of Papers*, Oct. 1999, pp. 199–202.

[112] M. Vidal, A. Bulling, and H. Gellersen, "Pursuits: Spontaneous Interaction with Displays Based on Smooth Pursuit Eye Movement and Moving Targets", in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '13, New York, NY, USA: ACM, 2013, pp. 439–448.

[113] H. Wallop, *CES 2010: Breath-controlled mobile phones to be made? Telegraph.co.uk. Accessed: 2016-07-14. 2010.* 2010.

[114] H. Wen, J. Ramos Rojas, and A. K. Dey, "Serendipity: Finger Gesture Recognition Using an Off-the-Shelf Smartwatch", in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16, New York, NY, USA: ACM, 2016, pp. 3847–3851.

[115] J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li, "Gesture Recognition with a 3-D Accelerometer", in *Ubiquitous Intelligence and Computing*, ser. Lecture Notes in Computer Science 5585, D. Zhang, M. Portmann, A.-H. Tan, and J. Indulska, Eds., Springer Berlin Heidelberg, Jan. 2009, pp. 25–38.

[116] H. Xia, T. Grossman, and G. Fitzmaurice, "NanoStylus: Enhancing Input on Ultra-Small Displays with a Finger-Mounted Stylus", in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ser. UIST '15, New York, NY, USA: ACM, 2015, pp. 447–456.

[117] R. Xiao, G. Laput, and C. Harrison, "Expanding the Input Expressivity of Smartwatches with Mechanical Pan, Twist, Tilt and Click", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14, New York, NY, USA: ACM, 2014, pp. 193–196.

[118] X.-D. Yang, T. Grossman, D. Wigdor, and G. Fitzmaurice, "Magic Finger: Always-available Input Through Finger Instrumentation", in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '12, New York, NY, USA: ACM, 2012, pp. 147–156.

[119] Y.-Y. Yeh and C. D. Wickens, "Dissociation of performance and subjective measures of workload", *Human Factors*, vol. 30, no. 1, pp. 111–120, 1988.

[120] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, "The HTK Book (for HTK Version 3.2). Cambridge University Engineering Department", *Cambridge, UK, December*, 2002.

[121] S. Zhai, P. O. Kristensson, P. Gong, M. Greiner, S. A. Peng, L. M. Liu, and A. Dunnigan, "Shapewriter on the iPhone: From the Laboratory to the Real World", in *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '09, New York, NY, USA: ACM, 2009, pp. 2667–2670.

[122] C. Zhang, A. Bedri, G. Reyes, B. Bercik, O. T. Inan, T. E. Starner, and G. D. Abowd, "TapSkin: Recognizing On-Skin Input for Smartwatches", in *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces*, ser. ISS '16, New York, NY, USA: ACM, 2016, pp. 13–22.

[123] Y. Zhang and C. Harrison, "Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition", in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ser. UIST '15, New York, NY, USA: ACM, 2015, pp. 167–173.

VITA

Gabriel Reyes earned his Bachelor of Science degree in Electrical Engineering (minors in Business Administration and East Asian Languages) from the University of Florida in 2008. He also earned a Master of Science in Electrical Engineering (minor in Computer Engineering) and a Master of International Business (minor in Global Management) from the University of Florida in 2011. During his time at the University of Florida, he was advised by Electrical & Computer Engineering Professor Jenshan Lin and Changzhi Li in the Radio Frequency Circuits & Systems Research Laboratory.

Beginning in 2011, he joined the School of Interactive Computing at Georgia Institute of Technology, where he was co-advised by Professor W. Keith Edwards in the Pixi Lab and Professor Gregory D. Abowd in the Ubicomp Group. He also worked closely with Professor Thad Starner in the Contextual Computing Group. In 2017, he graduated with a Doctor of Philosophy in Computer Science with a focus in Human-Computer Interaction.

His research interests lie at the intersection of human-computer interaction, ubiquitous and wearable computing, embedded systems, and sensor technologies. Specifically, his work focuses on building devices, systems, and algorithms that enable forms of gestural input, novel interactions, medical sensing, and activity recognition. He leverages sensors, applied machine learning, signal processing, computer vision, and physical prototyping to imagine and build always-available, real-time, mobile and wearable interactive systems. Gabriel is a recipient of the Google PhD Fellowship in Wearable Computing.